DRAGON USER

The independent Dragon magazine

August 1986

Contents

Editorial

SUBSCRIPTIONS are still pouring into Little Newport Street, to the delight of everyone except poor Anne Marie, who has to enter all the details. Suggestions and opinions have also been pouring in. A few people who are short of money are worried about losing touch. Help your fellow Dragon-users to stay in touch by carrying the latest issue in your back pocket and whipping it out wherever Dragons gather — back issues will be available, and new subscribers are welcome at any time.

The special offer of £12 for a year's subscription continues this month, so if you know someone who missed the June issue — draw it to their attention.

This month DU begins a new series from technical maestro Brian Cadge on language alternatives to Basic, starting with Pascal, and we re-join Orbaum and Campbell in their epic trip into machine code, with a double helping to make up for last month's dearth.

We depend on your feedback, so write and tell us what would be useful.

Telephone number
(All departments)
01-437 4343

Editor
HELEN ARMSTRONG

Production Editor
BARBORA HÁJEK

Associate Editor
JOHN COOK

Editorial Secretary
ANNE MARIE O'DWYER

Advertisement Manager
SIMON LANGSTON

Administration
GERALDINE SMYTH

Managing Editor
PETER WORLOCK

Publishing Director
JENNY IRELAND

Subscriptions UK £14 for 12 issues
Overseas (surface) £20 for 12 issues
ISSN 0265-0177. Telex: 296275
Dragon User, 12/13 Little Newport Street, London WC2H 7PP
US address: c/o Business Press International, 205 East 42nd St, New York, NY 10017
Published by Sunshine Books, Scot Press Ltd. ©Sunshine Books 1986
Typesetting by Chesham Press, Chesham, Bucks. Printed by Hedley Bros, Ashford, kent.
Registered at the Post Office as a newspaper
Dragon and its logo are trademarks of Eurohard Ltd

How to submit articles

The quality of the material we can publish in *Dragon User* each month will, to a very great extent depend on the quality of the discoveries that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to *Dragon User* for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, whenever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy. If you want to have your program returned you must include a stamped, addressed envelope.

Letters

This is the chance to air your views — send your tips, compliments and complaints to Letters Page, *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

More to come

I WAS disappointed to hear that the mag is going subscription only, but I understand your reasoning, and it's better to pay £12 in one go than not to buy DU at all!

I hope that your promise to 'pack even more in' means more pages.

Would you please give consideration to articles on the following: Use of Dragondos (the manual is just about useless); adding a second disk drive using one of the many cheap drives now available; OS9; Flex (especially an article by Compusense); interfacing (to robots etc) and more technical details on the hardware and circuitry.

D. J. Barham
649 Obelisk Rise
New Boughton Green
Northampton NN2 8TG

PS Any idea where I can get hold of a copy of the OS9 operating system? I'm getting desperate.

We would like some more news on the OS9 system ourselves. Can anyone assist?

Useful programs

OVER the years I have developed a number of short Utility programs which I have found very useful. They include an easy tape load and search program, a program to auto-run and partially protect both Basic and M/C programs — this includes a load screen designer, a program to enable you to make copies of your own autorun programs, a Merge program, for easy merging of Basic programs, a List program which by listing one line at a time and providing easy movement up and down etc, make listing easy, and a full 6809 disassembler for the Dragon.

If any of your readers would like a copy of these programs plus instruction sheets I would be glad to send them one. They should either send a tape, sae and 17p stamp to cover photocopying or their address plus £1 to cover everything.

Tony Sewell
44 Balfour Road
Walmer
Deal
Kent

Add on Colour

AFTER reading the review of the Dragon Plus in the January '86 DU, I had the impression that unless one has a disk drive and wants to use Flex this expansion is of no use. Is this correct? In relation to graphics, does anyone make an add on that will give colour in PMode 4? The Dragon Plus gives 80×24 display, what effect does this have on the graphics?

J. E. Smith
35 Bewick Crescent
Newton Aycliffe
Co Durham DL5 5LJ

Compusense are the agents for Flex only, so they don't sell OS9. They don't do cassette based systems either, so their own software is written around the requirements of Flex. There is no software on the board itself — if you can obtain suitable software control, you can hook it up to any system. The Dragon Plus does not alter the graphics capabilities, being a text-only display.

Well Done!

I AM writing to DU to say thank you to all the staff who helped get many a delayed copy of *Total Eclipse*. Without their support many readers might not have known who to contact. I hope you print this, so that people can see, that without DU, the Dragon just could not survive.

S. Bloomfield
14 Church Road
Great Stukeley
Huntingdon
Cambs PE17 5AL

New Shape

IN RESPONSE to I. M. Macdonald's query as to whether or not the Dragon 32's cursor could be made to change its shape, I have written this small routine to do just that:

```
10 FOR X=&H200 TO &H215
20 READ A$
30 POKEX,VAL("&H"+A$)
40 NEXTX
50 DATA 8E, 02, 0C, BF, 01,
   6B, 86, 7E, B7, 01, 6A, 39,
   34, 12, 9E, 88, 86, 2A, A7,
   84, 35, 92
60 EXEC 512
```

Enter the program and RUN it. As it stands, the program changes the cursor to an asterisk, but can be altered by typing: POKE &H211,ASC ("your character").

Stuart James
34 Gainsborough Drive
Perton
Wolverhampton
WV6 7NR

Dragon Society

I WOULD like to let Dragon User readers know about a users group I have started. It is called the Dragon and Co/Co Users Society. Members will receive a quarterly newsletter that offers hints and tips, answers to members' queries, program listings and a chance for members to get in touch with each other. Anyone joining will receive a machine code utility, to auto-run their own programs.

Membership is £1.80 a year and further information can be obtained from me at the address given.

Kevin Coleman
164 Elms Vale Road
Dover
Kent CT17 9PN

Correct Tips

IN THE April edition under "More Tips" there were two mistakes. The first line should read:

```
20 FOR A+1 TO 18
```

The second was line 40 and should read:

```
40 SWRITE 1, 20, A, A$, B$
```

R. Bailey (G4PPP)
52 Princess St
Chase Terrace
Staffs WS7 8JN

Amateur Radio

I AM writing to advise you that I am organising a Dragon Amateur Radio User Group. This group will cater for licensed and listener radio amateurs and will explore the capabilities of the Dragon (both 32 and 64) in the field of amateur radio to the full. Having seen the letters following Martin Vernon's letter, I would like to say that this group will be specialised and devoted solely to amateur radio with the Dragon. I would be grateful if you would publish this short letter and anyone interested should contact me at the address below. We have already published two newsletters and the third is due out in July. The subs are £3 per annum.

Roger Woods GW8XAN
20 Heol-ap-pryce
Yorkdale
Beddau
Pontypridd
Mid-Glam
CF38 25H

Back Copies

WHERE and how can I buy older back issues of DU? I mean from 1984-1985.

Lots of people

We normally only have back numbers for the last six months, although if you're lucky you can find a few older issues. We ask £1.25 post paid per issue.

We can supply photocopied articles from some earlier issues for £1 per article, irrespective of length.

If we can't supply the issue or article you ask for, your order and money will be returned.

Dragon User People's Chart

Results July 1986

THE BARONS are back — *Juxtaposition* takes the top position again after two months at number two. We have a new climber — *Bean Stalker*, and *Jet Set Willy* raises his head again.

This month's anagrams plumped for straight-down-the-line sincerity. Howsis, from Phillip Taylor? "Please choose me as I do hate cons." Would we dare? And we spy a new trend among the Speccy bashers — our first crop of anti-Amstrad sentiments! We can't print the current samples, regrettably, but it's a sure sign of the all-in-one wonder's world success. We prefer "Be beastly — get a Dragon!" from Deborah Dean, and Simon Scoltock's cry from the heart "I should be revising". But this month's prize goes to Jason Lee of Huddersfield for "Snooker Loopy mad are we, the Dragon just like me". So why didn't you vote for it, Simon?

Vote, vote, every month, for your current favourite five Dragon programs. They can be anything . . . games, utilities, what you like. Anything except the tape head cleaner. Write them in order of preference on the form printed here (or on your own paper).

Don't say we get something for nothing — £25-worth of Microdeal software will be given away to our favourite phrase, constructed from the letter in your own top three titles (use as many as you like). You don't have to include a phrase to vote, but you won't win unless you have a go!

1 Juxtaposition.................... **(Wintersoft)**
2 Bean Stalker.................. **(Micro Vision)**
3 Shocktrooper .... **(Microdeal)**
4 Moon Cresta........................ **(Incentive)**
5 Jet Set Willy.......... **(Software Projects)**

Chart Seven

Voting for Chart No. 7 closes at 1pm on Friday, 15th August, 1986. Entries received after that time will not be eligible for inclusion in that month's voting. The editor's decision is final. Only one entry per individual per month will be allowed.

My top 5: Voting Month 7

1. ....
2. ....
3. ....
4. ....
5. ....

Name ....

Address......................................

....

....

....

....

My phrase is: ....

News desk

If you have any new products for the Dragon — software or hardware — ring the News Desk on 01-437 4343

Sun starts to shine for Eclipse?

TOTAL ECLIPSE, the vast space trader game which ran into problems on its maiden voyage, looks as though it's coming out from behind the cloud at last.

Eclipse-Fenmar chief Trevor Davies believes that every customer who contacted the company has now been sent the mark 1.3 version of the game, free of the elusive bugs which had stopped play. "To the best of my knowledge, there's nobody out there who hasn't received a game who ordered one. The Birmingham Consumer Services Department put one more chap through to me this week."

"We were quite surprised at the public's reaction. They have stuck with us. People who were really unhappy at the time have contacted us to say how pleased they are."

"It has taken time to deal with mail because we had to let some of our staff to go, and the office is not manned full time. We also lost some mail because of this. It has really knocked us for six. I put money into it and went in over the top."

"People have asked us about our next game. We are contemplating another game, but it will take us a while to get back on our feet."

Birmingham Consumer Services confirm that Eclipse "apparently have it sorted out. They're not fly-by-nights, and I know they've been making efforts. They are still based at the same address."

The address, for anyone with further queries, is Eclipse-Fenmar, Suite 10, 4 Orphanage Road, Birmingham B24 9HS.

Total Eclipse, tipped as game of the year by DU's reviewer before the troubles, is available from Eclipse-Fenmar at £9.95, or win one in this month's Gordon Lee puzzle.

Prize Books in lieu

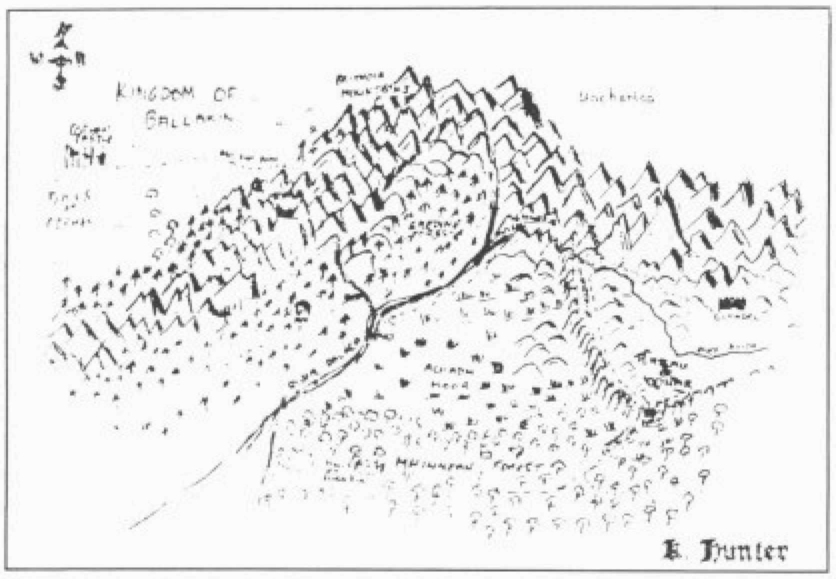
MELBOURNE HOUSE, who have been making efforts to find a few of the last remaining *Enter The Dragon* tapes for Dragon User's outstanding January prizewinners, have contacted us with the sad news that nobody they have spoken to has a spare tape under the counter.

Melbourne House send their apologies to all concerned, and those winners who did not receive tapes will be getting books in lieu.

Text Adventure

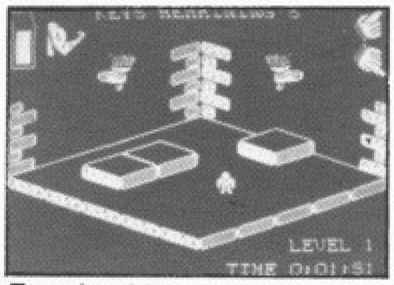
A NEW classic-style text adventure running under the FLEX DOS is on its way. The *Curse of Comarr* boasts 100 locations and 20-plus characters in 47k of machine code plus 10k in the utility command space on disc. The program has a large vocabulary, and FLEX commands can be used during the course of the game.

Look out for a review from Roy Coates soon. *The Curse of Comarr* can be ordered from K. Hunter, 46 Greenhill Road, Elton, Bury, Lancs BL8 2LL, for £10 post paid.



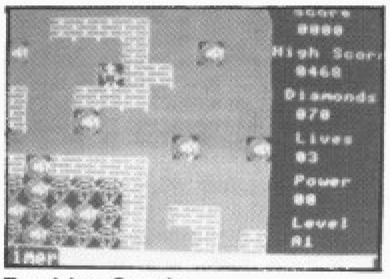
Radio Amateurs impress Blaby

BLABY have two new games with us now and another on the way, so look out for reviews next month. Temple of Doom, Boulder Crash and Trun are the names to watch out for.



Temple of Doom

"The Dragon show in Manchester was very successful for us" says John Bailiss. "A comparatively small number of people came, but they're all serious Dragon people." John was impressed by some practical demonstrations of Dragon programmes being used by Radio Amateurs to send pictures by RTTY.



Boulder Crash

Dragon User would like more information on Amateur Radio software for the Dragon and Tandy micros, so if you have anything to say on the subject, drop us a line.

68 Micro Group

ANY solitary hardened Dragon or Tandy CoCo hackers might be interested to hear of the 68 Micro Group — an established national club dedicated to users of all 68xx systems.

Membership includes a nicely put together A5-type journal (named 68 Microcosm), access to a library of assorted software (much of it FLEX and OS-9) and, of course, contact with other enthusiastic users. Perusing the April edition of 68 Microcosm showed it to be 30 pages packed with assembler and circuit diagrams, special hardware offers (how about 46 quid for a Dragon Prestel cartridge + a Modem 2000, direct Modem House?) — with four pages directly dedicated to the Dragon itself.

Although perhaps not for raw beginners, this ambitious group (they plan to hold nationwide monthly meetings, where there is the support) is certainly worth investigating. For membership details, write to Jim Turner at 63 Millais Road, London, E11.

Xmas Show

MICRODEAL'S 6809 CHRISTMAS SHOW will be held on 22 November at the Royal Horticultural Halls, Westminster, London. For more details phone Jenny Pope on 0726 6820. The Royal Horticultural Halls are within easy walking distance of Victoria Station, British Rail, and Victoria and St James Park underground stations.

Microdeal have two new games in the pipeline: Cuthbert and the Golden Chalice, and Tanglewood. No release date yet, but we hope to be reviewing these as soon as they become available.

Communication

Send in you questions, requests, and pleas to Communication, Dragon User, 12-13 Little Newport Street, London WC2

Problem: I want to meet any Dragon owners in the Southampton area, as I own a Dragon 32 but don't know anyone else who does.
Enquirer: Jason Coombes 52 Springfield Avenue, Holbury, Southampton SO4 1LP.

Problem: Wanted "text on hi-res screen" MC routine for adventure program under Cumana DOS. Can anyone help?
Enquirer: J. D. Law-Green, 1 Whitelands, Rawdon, Leeds, W. Yorks LS19 6BW.

Problem: I have Cuthbert in Space and I used the pokes (D.U. March 1986) but I still get only four lives. Help. Also I would like a Dragon owner penfriend, preferably in the Gloucester area.
Enquirer: Paul Palmer, 15 Underhill Road, Matson, Gloucester GL4 9HB.

Problem: I have a Dragon 32 with a Star Gemini 10× printer. I cannot follow the instructions for printing graphics. They read as follows: FORMAT: CHR$(27) CHR$(75) CHR$(n1) CHR$(n2) CHR$(m1) CHR$(m2) . . . The number of columns to be printed is given by n1+256*n2. There must be n1+256*n2 characters following n2. So far I have failed to work out any combination of figures to give me a print out. Could anyone explain? Preferably with an example.
Enquirer: D. S. Henserson, Cumbria, Larners Drift, Toftwood, Dereham, Norfolk NR19 1LE.

Problem: I have a Dragon 32, and have some difficulty writing a program for dumping screen graphics, graphs, etc. to hard copy. I have a Brother 1009 printer. Any help would be most welcome.
Enquirer: J. Airey, 1 Sandford Avenue, Gosport, Hants PO12 2SS.

Problem: I have written a program that scrolls the screen to the left. The sprite (a spaceship) which should move up and down, left and right, using peek keys, leaves a tail behind. Could anyone give me a program to tip in Basic to relieve this problem?
Enquirer: H. Woodcock, 131 Penistone Road, Grenoside, Sheffield, S30 3QH.

Problem: I have owned a Dragon 64 and disk drive for a year with no problems. Recently an intermittent fault has developed in my system. The symptoms are that when I type any DOS command I get an error, eg DIR or LOAD an RF error and BOOT for OS9 an NR error. The disk motor is working. I have checked my disks on another system and they are all OK. Could anyone let me know if they have had a similar problem, and do they suspect the DOS or the drive?
Enquirer: Peter Duncombe, 8 Arden Grove, Harpenden, Herts AL5 4SJ.

Communication

Stuck for a routine? Need some obscure equipment? Feeling cut off? Fear not — someone, somewhere can help you! Write down your problem on the coupon below (make it as brief and legible as possible) together with your name and address and send it to Communication, Dragon User, 12/13 Little Newport Street, London WC2H 7PP. We'll publish it as soon as we can — meanwhile, maybe there's someone you can help this month!

Problem..
....
....
Name..
Address..
....

Dragonsoft

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

Bowled out

Program: *Superbowl*
Supplier: Computape (Game by Cable)
Price: £2.99

LO, life in the happy land across the waters had fallen upon even harder times than when last we told of it. With the general disenchantment among the little folk with the good witch Cablette and her foiled blessing in *Zak's Son* the little folk had put away their Dragons and plugged a normal aerial in the back of their little televisions and a new craze had sweeped across the land.

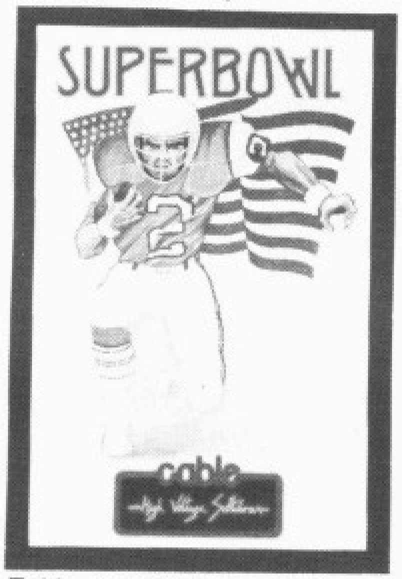
For verily hadst the people discovered Channel 4 and found how very different it was and exciting with programs of alternative arts, and minority groups having their own programs (which did raise the hearts of the hobbits in the land who up until then had been given a fairly hard Press by Melbourne House), and alternative sports.

They were Basketball and American football. But soon the craze didst die out for basketball, as the little folk had not the height to get the ball in the basket and so American Football became the craze.

And high in his castle among the dark mountains the evil wizard didst laugh and count his money a lot and commit all of the seven deadly sins all at once and then laugh a lot more for he had finally removed all the Dragons from the land (for lo he didst hate computers as much as happiness).

So, as you can see, things looked pretty damn bad for the little people. But then the good witch Cablette didst crack it and released a game based upon American Football, which shows a top view of players running up and down the pitch and is generally a great idea.

Sadly though, the good witch had had much of her energy sapped by the defeat of



Zak's son by the evil wizard and so all she could produce was a game with nice smooth graphics but little else.

And lo, once again the little people didst send off for it and get all their Dragons out, and load it up, and play it for a couple of nanoseconds, and get bored out of their wits with it, and put their Dragons away again, and throw away the game, and go back to watching alternative arts programs.

And the good witch Cablette was banished from the land for letting the people down again. And the Dragons didst never come out again. And high in his castle the evil wizard laughed that all this had gone so well for him when he'd really had nothing to do with it. And he didst come forward and crown himself king, and his name was Cliff Sinclove.

And the morals of this tale are firstly that if no one gets their A into G about producing software then computers die, and secondly those who bring out two rubbishy games in a row can't complain about two cynical reviews.

Jason Orbaum

Old zoner

Program: *Lazer Zone*
Supplier: Microdeal
Price: (See text)

A FEW YEARS ago, when I was very young, I went round to my mate's house to see a couple of new games he had acquired for his VIC-20. One of them was *Lazer Zone*. We played it all evening and I went home in incredible envy of this game.

I can't remember why I loved this game so passionately, but it's still a good game, only let down by the graphics on the Dragon! It puts you on two axis of a grid with objects coming at the two axis. On each axis you have a gun and the idea is to blast anything that moves out of the sky (or grid). Up and down on the joystick controls the gun on the vertical axis, and left and right correspondingly move the gun on the horizontal axis. And that's about it, save a couple of rather nifty touches that raise this game from the mediocracy.

The first is that by using diagonal movement it is possible to pick an alien that has landed on a certain axis off with the other gun (does that make sense?) This is a very difficult manoeuvre to execute involving a lot of practice to prevent suicide!

The second is that it is possible to play with two people in co-operation, one on each axis, and it's here that the game comes into its own.

The sound's great, the graphics are all right (although I prefer black and white) and it plays fine. I have given it a rating of three but I don't know whether I found the game a let down because it has not translated well to the Dragon, or because after a very long wait I had built up my expectations too highly, so it may be worth four.

The game is well worth a look if you can pick it up cheap somewhere (it's often going cheap at the shows) and you may find the initial passion for it that I shared. But, be warned, this is not an easy game!!

Jason Orbaum

Hooked

Program: *Ruby Robba*
Supplier: Blaby
Price: £1.99

IT HAD BEEN a hard day, outside the rain was pouring like it would never stop. God knows how long I'd been on this case. I thought it would never end. Armed guards, snakes, and rubies. They all swam round my head like it was some sort of aquarium.

I threw on my coat, missed, tried again, missed again, and then put it on properly. I'd never managed to learn that trick. I sighed.

She was there when I opened the door. A real picture of beauty. "You've got to help me!" she said, "I can't stop playing *Ruby Robba*." I sighed. I knew the symptoms, my God, I had them too.

"You want a drink?"

No reply.

"I'll get you a drink," I said. I sighed. I seemed to be doing a lot of that recently. I fixed her a gin. But when I got back into the room I found that all that was left was a faint memory of her perfume and a cassette inlay on the floor. I read it. "The object of this game is to steal the precious ruby by breaking the complex defence system guarding it."

There they were, the same old instructions. I went to the cupboard and opened it. Out fell thirty thousand *Ruby Robba* cassette inlays. There were thirty thousand others out there who were addicted.

Then I found a lead. I decided to follow it. It led to a plug stuck in a socket on the wall. Devious. And typical of the enemy. Blaby. That one word struck so much terror into the hearts of so many people. Blaby. For some it meant cheap games. But for most it meant addiction. I decided to follow the lead the other way. . . . It led to the cassette recorder connected to my Dragon.

I sighed and loaded up the game. It loaded fast with no problems. The graphics were smooth and flicker free. The sound was great. I knew there was a connection between this and the *Boris the Bold* case that had so nearly cost me my life. I had to play. I sweated. I sighed. I sighed again for good.

I sighed. Maybe the case was unsolvable. Maybe the game was too addictive. I looked at the screen. The blue light of the portable was all that lit the apartment. I sighed, and settled down to another night's play.

Jason Orbaum

What a wonderful show!

***Roy Coates** at Ossett.*

THE SECOND of the John Penn Dragon shows was held at Ossett Town Hall last weekend the 31st May. At first thought this seemed to be an odd choice of venue although after consulting a map it proved to be well thought out as easy access is gained from both the M62 and M1 motorways. This was borne out by the fact that there were actually people waiting outside before the show started.

It was nice to see a few new faces as well as most of the regular suppliers of Dragon hardware and software in attendance with some marvellous bargains and a very large selection of both programs and accessories for the Dragon.

Firstly (and deservedly so), John Penn had a large and well manned stand offering unbelieveable discounts on a comprehensive range of Software which included both games and utilities on both cassette and disc. Compusense as always, were displaying their extensive range of both hardware and software, most of which centres around the FLEX and OS-9 operating systems, which, judging by the amount of people gathered around their stand was generating some real interest. Ted Opyrchal from Compusense was keen to stress that they are still very much dedicated to the Dragon and will be continuing to support the Dragon users as long as they exist.

Blaby (as usual) were very busy demonstrating their vast range of games software with two new releases on show, *Boulder Crash* and *The Temple of Doom* with the promise of more new titles to be released shortly.

Eclipse-Fenmar were again displaying their *Total Eclipse* program and now seem to have come through their initial bad publicity problems simply through having such a very high quality game. For those of you that have been brave enough to tackle *Total Eclipse*, I have been informed that there is a third universe to be released shortly which must make this game possibly the longest playing adventure ever written for the Dragon. Good old Peaksoft were again there to tempt us with a huge range of accessories on show including everything from replacement power supplies through disc drives to sweat shirts. Although they do not supply software for the Dragon they probably have the largest stock of accessories for the Dragon.

Computape, which is one Dragon company that seems to be rapidly expanding, has a most comprehensive range of software at ridiculously good prices and a running battle seemed to be constantly in progress to get near enough to their stand to buy something! I must confess that they had







many Dragon titles for sale which I for one had never heard of before so my wallet certainly went home relieved of some of it's burden!

Smithson Computing were occupied showing off their new monthly magazine for the Dragon called *Dragon Monthly*, (ingenious title eh?), as well as their *Electronic Author* and *Gordon Bennet* programs.

Grosvenor Software had an impressive display of goodies on show which included the 'Dream' range of programs which consist of assemblers/disassemblers, word processor etc, etc, as well as their products to keep the amateur radio enthusiasts happy with RTTY units, Slow-scan TV and the software to support these units as well as other software of specific interest to the amateur fraternity.

Microvision had *Beanstalker* on display as well as representing Incentive and Software Projects by displaying *Moon Cresta* and *Jet Set Willy*.

What made this show different from previous ones were the stands that weren't actually selling anything but which were demonstrating how they used the Dragon for a specific application. Three radio amateurs (GOA1Z, G4TQR and G6ZZE) had set up two Dragons at opposite ends of the demonstration hall and were communicating between them using a radio link on the 2 metre FM amateur radio band, with great success. What wasn't obvious was that this software is available to the general public, if you are interested contact Blaby's John Bailiss who is himself a licensed radio amateur (G1LTJ). The Dragon does seem to have become almost a 'cult' machine with radio amateurs all over the country and there are many groups who are sending software via radio on a regular basis. You do not have to be licensed to receive these transmissions but you will need a receiver capable of tuning in the VHF range at about 144Mhz.

The Verulamiun Museum from St Albans came to demonstrate how they have been using Dragons to assist in the excavating of a Roman cemetery in St Albans, the software that they have developed is very impressive and allows the accurate mapping of the graves to be excavated and also the comparison of various sites to be made by either displaying multiple maps on the graphics screen or by overlaying one on top of the other. Similar programs have also been written to allow for the precise measurement and comparison of the ceramic pots which are common to this type of site.

One of the stands had been taken by a young programmer who was exhibiting his software in the hope that it may be taken up by a software house. One of the packages that I looked at was the *Composer Companion* which co-resides with Microdeal's *Composer* package and allows musical data entry to be made via a musical stave as opposed to those awful DATA statements that make data entry and correction an appalling job. If you are interested, contact Jonathan Cartwright at Starship Software, 23 Tintern Road, Cheadle, Hulme, Cheshire SK8 7QF.

A lot of visitors to the show were bemoaning the fact that there is very little software available on disc. Having spent a tidy sum on a disc system it is rather galling when you find that you can't use it. Maybe we shall see more disc based software from now on?

From an exhibitor's point of view, one thing that struck me was that all the exhibitors seemed to know each other well and the atmosphere was a very friendly one as opposed to a competitive one. The relationship between them being more like a family instead of a business.

In conclusion, the show was a great success judging by the smiles, and each of the Distributors that I spoke to had the same thing to say, "What a wonderful show". If I have missed anyone out, my apologies, it hasn't been intentional but the show was busy and it was difficult to get around everyone.

Flag And Branch

*Part Five of our machine code series — **Jason Orbaum** waves the flags.*

HELLO, yes, it's me, alone again — this month talking to you about the CC flag and Branch instructions. You'll have to forgive me if I wander off the subject, it's just I'm missing Geoff and, if he's reading this, come back. And bring the disc drive with you!

First, a big thank you to a certain Mr Martyn J. Preston who writes to inform us all of where we can get the elusive Motorola specification sheet on the 6809. Apparently in the Hitachi *Semiconductor Data Book — 8/16 Bit Microcomputer* there are about 30 pages on the HD6809E and the HD6809 as well as data on the 68xx and 63xx microprocessors and 68xx peripheral chips (6821, 6840, 6850, etc). There is also, according to Martyn, some information on the HD68000. Martyn got his book from Farnell Electronic Components Limited, Canal Road, Leeds LS12 2TU. The stock number for the book is 171-360 and it should cost £7.50. Thanks again for that, Martyn.

So, on to the business in hand. This month, after last month's rather simplistic article we have aimed slightly higher. If you find the information left unexplained to a degree or you still find it too simple to read, please write and let us know at the usual magazine address.

Below is a dissection of the CC flag into its respective bits with descriptions of those immediately relevant.

The CC Register

: E : F : H : I : N : Z : V : C :

H: Half Carry
This bit is set (contains a value of 1) when the result of any mathematical calculation results in the fourth bit of the resulting byte being set. This will become clearer after next month's tutorial on additional instructions.

N: Negative
This is pretty obvious. The bit is set if the result of a mathematical operation should be negative.

Z: Parity
Set on equality, ie if the two elements of a CMP instruction are equal the Z bit will be set.

V: Overflow
Set if the result of an eight bit operation mathematical operation. This is the way that negative numbers are denoted in binary. Next month's rather lengthy article will explain both 2's complement and BCD denotation in binary. It was decided after last month it would be better to give them a miss this month and steer off the theory back into the commands and practice.

C: Carry
Set if the result of an eight bit operation causes the need for a ninth bit, ie 11111111B + 1B =100000000B. The result, as can be clearly seen, has nine bits. The ninth bit becomes a set carry bit in CC and the byte becomes 0B.

These then, for the moment, are the important bits in the CC flag. Let us now see their relevance to the branch instructions. The branch instructions covered here are not all of those in this month's table. However, they are the only ones you will need for now. I shall use the current situation with me and Geoff as an example for illustrative purposes.

BEQ: Geoff comes back if and only if it's on his terms.

BNE: Geoff comes back provided it's not on his terms.

This pair, as can be deduced, stand for Branch if Equal and Branch if Not Equal. they are used after arithmetic calculation (as are most of the branch instructions).

BLO: Branch if Lower: Geoff comes back if he agrees to drink less.

BLE: Branch if Less than or Equal to: Geoff comes back if he agrees to drink NO MORE than he did before.

BHI: Branch if Higher: Geoff comes back if he agrees to give me more rent than before.

BHS: Branch if Higher or Same: Geoff comes back if he agrees to give me NO LESS rent than before.

BRN: Branch Never: Geoff never comes back.

BRA: Branch Always: Geoff always comes back.

Quite quick and painful really, wasn't it! The idea is that you take this, look at the table of branch instructions, match them up, and then look at the following piece of code and work out at which lines the code will RTS for the numbers given.

```
 10 LDA #NN
 20 CMPA #0
 30 BEQ POINT1
 40 CMPA #50
 50 BHS POINT2
 60 CMPA #32
 70 BLO POINT3
 80 SUBA #32
 90 BEQ POINT4
100 BRN POINT5
110 RTS
120 POINT1 RTS
130 POINT2 CMPA #200
140 BLO POINT6
150 RTS
160 POINT3 RTS
170 POINT4 RTS
180 POINT5 RTS
190 POINT6 RTS
```

The NN in line 10 stands for a number picked from the following list. Work out where each takes the program:

100, 32, 33, 0, 232, 233, 200, 199, 50

The answers, respectively, are the following lines:

190, 170, 110, 120, 150, 150, 150, 190, 190

If you got the exercise correct then you can congratulate yourself on passing this small, but complex part of the course. And that really is it for this month. Next month Geoff gets back from holiday (yes, all that stuff about him leaving was a big joke, and boy, is he going to find it funny when he sees it in print!) and to celebrate we'll be presenting an extra long edition. So, for those of you who like to read up ahead of us next month the following interesting and varied topics will be covered:

(1) Assembler directives (maybe, we've been promising this one for so long now that it's almost fun to not do it each month).
(2) The differences between LBNE and BNE and other related topics, which leads nicely into
(3) Addressing modes.
(4) Arithmetic, complete with diagrams and tables.
(5) BCD and 2's Complement arithmetic.

All this and more that you will hardly believe! Or understand. (Cue lots of letters!!)

Branch Instructions

BCC — Branch on Carry Clear

Mnemonics: BCC & CBCC
Function: If C=O then branch to specific point
Addressing Mode: Relative

BCS — Branch on Carry Set

Mnemonics: BCS & CBCS
Function: If C=1 then branch to a specific point
Addressing Mode: Relative

BEQ — Branch on EQual

Mnemonics: BEQ & CBEQ
Function: If Z=1 then branch to a specific point
Addressing Mode: Relative

BGE — Branch on Greater than or Equal to

Mnemonics: BGE & LBGE
Function: If (N (XOR) V) = 0 then branch to specified point

Addressing Mode: Relative

BGT — Branch on Greater Than
Mnemonics: BGT & LBGT
Function: If Z and (N (XOR) V) = 0 then branch to specified point
Addressing Mode: Relative

BHI — Branch on HIgher
Mnemonics: BHI & LBHI
Function: If (C or Z) = 0 then branch to specified point
Addressing Mode: Relative

BHS — Branch on Higher or Same
Mnemonics: BHS & LBHS
Function: If C = 0 then branch to specified point
Addressing Mode: Relative

BLE — Branch on Less than or Equal to
Mnemonics: BLE & LBLE
Function: If Z or (N (XOR) V) = 1 then branch to specified point
Addressing Mode: Relative

BLO — Branch on LOwer
Mnemonics: BLO & LBLO
Function: If C = 1 then branch to specified point
Addressing Mode: Relative

BLS — Branch on Lower or Same
Mnemonics: BLS & LBLS
Function: If (C or Z) = 1 then branch to specified point
Addressing Mode: Relative

BLT — Branch on Less Than
Mnemonics: BLT & LBLT
Function: If (N (XOR) V) = 1 then branch to specified point
Addressing Mode: Relative

BMI — Branch on MInus
Mnemonics: BMI & LBMI
Function: If N = 1 then branch to specified point
Addressing Mode: Relative

BNE — Branch on Not Equal
Mnemonics: BNE & LBNE
Function: If Z = 0 then branch to specified point
Addressing Mode: Relative

BPL — Branch on Plus
Mnemonics: BPL & LBPL
Function: If N = 0 then branch to specified point
Addressing Mode: Relative

BRA — BRanch Always
Mnemonics: BRA & LBRA
Function: Branch to specified point
Addressing Mode: Relative

BRN — BRanch Never
Mnemonics: BRN & LBRN
Function: Branch nowhere ever! (This is only included for symmetry.)
Addressing Mode: Relative

BSR — Branch to SubRoutines
Mnemonics: BSR & LBSR
Function: Branch to specified address leaving present location on system stack S
Addressing Mode: Relative

BVC — Branch on oVerflow Clear
Mnemonics: BVC & LBVC
Function: If V = 0 then branch to specified point
Addressing Mode: Relative

BUS — Branch on oVerflow Set
Mnemonics: BUS & LBUS
Function: If V = 1 then branch to specified point
Addressing Mode: Relative

JMP — JuMP
Mnemonic: JMP
Function: Jump to specified point
Addressing Modes: Extended
Directed
Indexed

JSR — Jump to SubRoutine
Mnemonic: JSR
Function: Jump to subroutine at specified point leaving current address on system stack S
Addressing Modes: Extended
Directed
Indexed

Addressing Modes

*Part six follows fast, with **Geoff Campbell** on the spot.*

JASON presented a piece of prose designed to explain the intricacies of the various branch instructions, and I for one think it was about as clear as mud. Still it should become clear in time. As the brave man wrote that entire section all on his own, I thought it was time we had a column devoted to doing what we set out to do — ie to teach others to program a computer. To this end, I have chained him up outside and taken over. I will cover a few subjects related to the branch instruction, some more on computer arithmetic, and an introduction to the various addressing modes, or ways of accessing information, that the processor had.

First a subject that was at one time regarded as the most important in computer science, that of decimal representation of numbers. It is possible, and indeed most efficient, to work entirely in binary, but remember that other people will be using your programs when they are finished, and there are very few businessmen, shopkeepers, personnel managers (or, in short, people) who are fluent with the binary number system, so any numerical results from a program must be displayed in decimal. The most efficient method of doing this depends on the application of the program. For a program with a lot of calculations and little result display, it is most efficient to hold the numbers internally as binary, and convert them to decimal when they are displayed. This is quite straightforward, and we will present a routine to do so later.

There are other cases, though, where this will not be the most efficient method, in terms of speed if not storage. If a program is doing a lot of displaying of decimal numbers, but very few calculations, as in the vast number of databases currently in use, it may be easier to store the numbers as decimal. Yes, I know we are not supposed to be able to do that in a binary computer, but this is where we start to cheat slightly (but only slightly), and introduce a new concept called Binary Code Decimal, or BCD for the lazy ones among us. BCD is very simple in principle, and is in fact not dissimilar to hex in practice.

Binary code decimal

Just as, in hex, each nibble represents one digit between 0 and F, in BCD each nibble represents a digit between 0 and 9. It is therefore fairly easy to see that a single byte is limited to a maximum value of 99.

This is no problem, because we can easily string together as many bytes as we like. When it comes to displaying the number, it is fairly simple to mask off the relevant nibble, and display an ASCII character. We will be presenting an article in the not too far distant future with a collection of small but useful routines, and this will hopefully include a set of BCD arithmetic subroutines (if I get round to writing them . . .).

There is a slight variation on the BCD principle which I have never seen anyone else use, which is to have just one digit per byte. While this takes up twice the storage, it does allow for very fast addition, subtraction, and display, and the digits can be held as ASCII (30H to 39H) rather than straight binary, allowing the display routine to simply copy them directly to the screen. This is very useful for applications like game score displays, although not so good for general calculations.

Another problem with conversion from decimal to binary is that of negative numbers. Negative numbers are often taken for granted, but they are in fact one of the most abstract concepts employed by the human race. Abstract concepts and computers normally mix like oil and water, but for once there is a reasonable easy solution to the problem.

If we consider a single word within the computer (although the principle applies across the board), we would normally expect to be able to hold a number from 0 to 65535 (a number that will fairly soon be engraved on your memory!).

Negative numbers

If, on the other hand, we sacrifice the most significant bit (the left-hand one) to represent either positive (set to zero) or negative (set to one), we end up with a range from 32767 (7fffH or 0111111111111111b) to −32768 (8000H or 1000000000000000b). This we have already touched on, and is fairly straightforward. What is a little less obvious is how to convert from negative decimal to binary and back again. For example, how about −53? It is actually a fairly simple process. First, convert the number to binary, ignoring the minus sign. This gives 00110101b (using a byte 'cos it means less typing). Next find the one's complement of this number (in other words, convert the ones to zeros and vice versa. This is easiest done in assembler by exclusive oring with ffH), giving 11001010b. Then add one, giving, in this case, 11001011b. This, I sincerely hope, is the binary equivalent of −53. To convert back, the process is exactly opposite.

Now a quick jump back to the test of branch instructions. Normally, branches the method outlined above for negative number, using the byte following the branch as an addition to the current PC value, giving a range of 127 to −128 bytes (NB from the start of the FOLLOWING instruction). There is a special case, however, whereby the branch is prefixed by the letter L, making it a long branch. The range is now 32767 or −32768, or, in other words, the entire memory map. This applies to any branch instruction available. Do not lose any sleep over calculating lengths of offsets for branches, because (a) it will make no difference if a long branch is used for a jump of 10 bytes, except using up an extra byte of RAM, and (b) the assembler will pick out any short branches that should be long branches. As a rule of thumb, use short branches throughout, and change any that the assembler throws out to long branches.

Now, after that nice easy start something to get you thinking (maybe — I was not thinking when I wrote it). Addressing modes. Just those two words have been known to make strong men weep, although it is in fact a fairly simple subject. I will cover the basics this month, and get more complex as and when we use them in routines.

Addressing modes

The addressing mode of an instruction is used by the 6809 to determine where to pick up or place the data for that instruction. The 6809 has a larger and more complex range of these modes than almost any other chip, barring some of the bigger sixteen and thirty-two bit monsters. I will cover each mode and its uses, although not all instructions can be used for all addressing modes. As we cover each instruction, the range of available addressing for that instruction will be given. The addressing modes are as follows:

Inherent Not an addressing mode as such, this means that all necessary information is included within the instruction itself. This covers instructions like INCA, which adds one to (increments) the A register.

Immediate in this case, the required data is taken from the byte (or word) following the op-code. This has the advantage of speed of execution, as the source address has already been calculated, and is in the PC register. This is common to most other processors, but, as we will cover later, the 6809 is unique in allowing the programmer to access these address calculations, and use them to produce TOTALLY relocable code and data (try that on your 6502!!). This can be fairly complex, so we will devote an article to it. In assembler source code, immediate data is always prefixed by the # symbol, as in ADDA #10, which adds ten to the A register.

Extended (or absolute) Possibly the most commonly used mode, this uses the word following the op-code as an address from which to gather the data (or as an address to which to write the data). In the source code, this is shown as a number with no prefix, or, more commonly, as a label. (For example, LDA 32767 would load the A register with the value stored at address 32767, but LDA SCORE is much clearer, SCORE having been previously defined by use of one of the assembler directives that we might cover next month.

Direct This is much the same as Extended, but uses a single byte following the op-code as the low byte of the address, and the contents of the DP register as the high byte. This is quicker to execute, and takes up a byte less storage, making it ideal for applications where there is a lot of data in a 256 byte area. Can be tricky making sure the DP register has the right value in without wasting more store than you save.

Relative Used (to the best of my knowledge) exclusively for branches, jumps and calls, relative addressing uses the contents of the byte or word following the op-code, plus the contents of the PC register, to calculate the address, which is normally then transferred back into the PC register. Again, this can be used to make code totally relocatable, but more of that later. At the source code level, this simply means that all that is specified is the target address, and the assembler will calculate the offset needed.

Indexed Again, with indexed addressing, the 6809 stands head and shoulders above a lot of other processors, in that it has two sixteen bit index registers, allowing access to the entire memory map without having to worry about base addresses. With this mode, the processor calculates the address form the word following the op-code, plus the contents of the specified index register (either X or Y). For example, LDA 1000, X will, if the X register contains 24, load the A register with the contents of location 1024, or the first byte on the text screen. This in itself is very handy, but there's more! If the register name is preceded or followed by either one or two pluses or minuses, the processor will use either auto increment or auto decrement modes, which means that, for example, LDA 1000,X+ will, if X contains 24 as previously, load the first byte of the text screen into A, then increment the X register, leaving it containing 25. Conversely, LDA 1000,+X will increment the X register first, loading A with the contents of the second byte on the text screen, and the X register containing 25. This is incredibly useful for accessing tables of information, clearing screens, and about a million and one other things.

Indirect When an indirect instruction is issued, the target (or source) address is the contents of the address contained in the word following the op-code. This is useful for things like tables of jump vectors. For example, a program might display a menu of options, accept a number for the user, multiply it by two (as each address takes up two bytes) placing the result in the X register, and then use an instruction like CALL (JMPVCT,X). Note the combination of addressing modes here. This is actually indexed indirect, and the combination of modes can get quite stunning, as it is also possible to use auto increment et al. Indirect on its own is not much use, although it is possible to find situations where it could be used. However, for each such situation, it is generally possible to find a more efficient method of gaining the same results.

Well, that about wraps it up. Sorry about the lack of assembler directives (again!) but they are coming soon. Next month, among the excuses for the lack of assembler directives, we (yes, we — Jason will be working again if I can think up a dire enough threat) will cover some more computer arithmetic.

Dragon dialects

***Brian Cadge** grabs his phrasebook and learns to parlez Pascal.*

IN THIS new series of articles we will be taking a look a some of the various languages available for the Dragon computers as alternatives to the built-in Basic.

This month we start off by looking at the language Pascal. For the purposes of this article I used Lucidata Pascal from Compusense which runs under the Flex operating system. Pascal was developed as a general purpose educational language by Nik Wirth in the late 1960's. Wirth intended Pascal to be used to teach structured systematic programming and hence the basis of the language is tailored to this style of programming.

The "Basic " approach to programming — that is given a problem, sit down at the keyboard and write a program, attempts to solve the problem by a mixture of inspiration and a lot of trial and error. While this approach can work for relatively very simple problems, it is entirely inadequate for finding solutions to programs of any real complexity. What is needed is a systematic approach to the problem, breaking it down into smaller and smaller steps until each step has a straight forward programming solution. This, basically, is the theory behind "structured programming".

Before diving into the realms of the language, take a look at the (very simple) complete Pascal program shown in **figure 1**.

```
PROGRAM factors (INPUT,OUTPUT);

VAR number, divisor, numofdivs : INTEGER;

BEGIN
  WRITE('Enter a number between 2 and 999: '); READ(number),
  WRITELN; WRITELN('Factors of ',number:4,' are:');
  numofdivs := Ø;
  divisor := Ø;
  WHILE divisor <= number DIV 2 DO
    BEGIN
      IF number MOD divisor = Ø
      THEN BEGIN
             WRITELN(divisor:3, number DIV divisor:4);
             numofdivs := numofdivs+1
           END;
      divisor := divisor+1
    END;
  WRITELN;
  IF numofdivs > Ø
  THEN WRITELN(numofdivs:4,' divisors found')
  ELSE WRITELN('No divisors found - ',number:3,' is prime')
END.
```

Figure 1: A simple Pascal program.

Basic structure

This shows the basic structure of a Pascal program. In this example, I have shown Pascal commands in uppercase and variables in lowercase for clarity — Pascal makes no distinction between upper and lowercase (although Lucidata Pascal does have the facility to do this).

The first line in a Pascal program always gives the name of the program (factors) optionally followed by the files to be used — here only the default keyboard and screen are used (Input and Output). Following this come constant, type, variable, procedure and function definitions. In the example program there are no constants, procedures or functions so only the variables used need be declared after the "VAR" command. Variable declaration before use is an important element of structured programming, implicit declaration is not allowed. We shall see later that there is a lot more to Pascal variables than it at first may seem.

Programs are made up of "blocks" of code — the whole program is the outer, top level block, next come procedures and functions, followed by procedures within procedures and so on. The lowest level block is an actual series of statements enclosed between the keywords "BEGIN" and "END". Pascal does not use line numbers and the semicolons are only used to separate commands for the compiler (they are not the same as colons in Basic). Therefore, with an IF-THEN for example, if more than one command follows the THEN part then they must be enclosed in BEGIN-END as shown in the Factors program.

Block nesting can be taken to any reasonable level, in this program there are only three levels — the main program block, the WHILE loop block and the IF-THEN block. Blocks can be thought of as representing the simplication of a problem into smaller and smaller parts.

Anything to declare?

Pascal is an example of a "strongly typed" language. What this means is that all variables must be declared together with their type before being used. Operations that can be performed on one type cannot be performed on another. Special exceptions to this are the so-called "overloaded" operators (such as "+" and "*") which can operate on a variety of types (such as integer and floating point).

Basic has only two built-in types, these are numeric (floating point) and string (character). Pascal has 4 simple types built-in, these are "interger" (16-bit numbers), "real" (floating point), "char" (single ascii character), and "boolean" (true or false.

Lucidata Pascal also has the additional simple types "byte" for byte integers and "alfa" for a string of eight characters. For efficient programs it is obviously necessary to use the most appropriate "type" of variable, integer arithmetic is a lot faster than using floating point for example.

Define your types

As well as the simple built-in types, Pascal allows you to define your own types to a limited extent. Often it may be necessary to deal with entities in programs (such as colours for example). In Basic we might use a series of variables assigned; RED=1: YELLOW=2: GREEN=3 and so on. Then within the program we can say; IF PAIN=RED THEN . . . This is an example

```
CONST maxnamelen = 1Ø;
      maxorders = 25Ø;

TYPE name = PACKED ARRAY [1..maxnamelen] OF CHAR;
     order = RECORD
               customer : name;
               partno, quantity : INTEGER;
               price : REAL;
               instock : BOOLEAN
             END;
     daybook = ARRAY [1..maxorders] OF order;

VAR todaysorders : daybook;
    ordfile : FILE OF order;
```

Figure 2: Pascal type definitions

of an "enumerated" type in Pascal. The equivalent would be declared as ;

```
TYPE COLOURS = (RED,
YELLOW, GREEN);
VAR PAINT : COLOURS;
```

The advantage in Pascal is that the variable "paint" can only take the values red, yellow and green and not just any numeric value as in Basic. Also you are prevented from performing arithmetic on enumerated types.

Record definitions

Pascal, like Basic, has multidimension arrays of any type. Unlike Basic the array is not the only data structure available to us. One of the most powerful features of Pascal is its "record" definitions. A record is a data structure consisting of a fixed number of components of various types. For example, if we needed to deal with "customer orders" which consisted of customer name, part number, quantity, price etc. Then in Basic the only solution would be several arrays such as NAME$, PART, QUANTITY etc. In Pascal a record type could be defined followed by an array of these records. **Figure 2** is an example of this type of definition, which also shows the use of constants in definitions.

"Order" is defined as a record type which consists of the elements mentioned previously. The type "Daybook" is then defined as an array of order records. Note that type definitions do not declare variables, only types of variables — it is then necessary to declare a variable such as "todaysorders" of the type "daybook". "Ordfile" declares a filetype consisting of orders so that todaysorders may be read and written to disk as whole records.

To access a particular field of a record we use, for example:
todaysorders[5].partno := 88;
That is, the record name followed by a fullstop, followed by the field name. some versions of Pascal allow the following type of command:

WITH todaysorders[5] DO BEGIN partno:=88; price:=1.08 END;

This saves the programmer from having to type todays orders[5] before each field name of a particular record. Although this is a very useful function, Lucidata pascal does not support it.

As mentioned earlier, Pascal is a block structured language, hence procedures may be declared and may be nested. The Basic "GOSUB" can be thought of as a very simple equivalent to the Pascal procedure. I will only deal with procedures here, but Pascal functions can be thought of as procedures which return values — procedures are called as new commands, functions are called as expressions; X:=F(S).

In Basic, a variable may be used at any point in the program and there is only ever one "version" of a particular variable. In Pascal, variables (or more formally "identifiers") have what's known as "scope", "binding" and "environments". A procedure may use its own variable which are not accessable by any other part of the program and whose values are not kept between calls to the procedure, these are known as "local" variables. Local variables may have the same name as variables in the main program ("global" variables) or as local variables in other procedures. Pascal allows recursion by the use of local variables — a new version of the variables is instantiated for each recursive call of the procedure.

The "scope" of a variable is that part of a program in which it may be accessed. When a variable name appears more than once, for example as a global variable and a local variable in a procedure, it is said to be "bound" to the current block. The variable exists only as long as the block in which it is declared is active. The environment of a block is the surrounding blocks in which it may access variables. The environments of all procedures include the main program (as this encloses all procedures), and also includes any procedures which enclose the procedure in the program text.

To clarify all this, take a look at **figure 3**, this is a completely useless Pascal program, but demonstrates the various scopes etc of variables.

Figure 3: Variable scopes and bindings.

```
PROGRAM xxx;

VAR main : INTEGER;

PROCEDURE p (param:INTEGER);
VAR localp : INTEGER;
    flagp : BOOLEAN;

  PROCEDURE q (main:BOOLEAN);
  VAR localq : INTEGER;
  BEGIN (* q *)
   main:=TRUE;
   localq:=1;
   localp:=10
  END;

BEGIN (* p *)
 q(flagp)
END;

BEGIN (* main program *)
 main:=1; p(main)
END.
```

Binding declaration

Here the variable "main" is declared as a global integer variable and as a local (boolean) variable for the procedure "Q" — it is declared as the parameter passed to "Q". Hence, when the procedure "Q" is entered, "main" receives a new "binding" to the boolean value and the global value of "main" cannot be accessed within "Q". When "Q" ends, "main" restores its binding to the global integer value.

"Localq" is a local variable of the procedure "Q" and so cannot be accessed by any other part of the program. Similarly, "Localp" is a local variable of procedure "P", but as procedure Q's environment includes procedure "P" (it is enclosed by it) it may access P's local variables. Notice that the main program can call procedure P, but cannot call procedure Q directly (it surrounds P, but does not directly surround Q).

All this binding, environments and scopes may seem confusing at first, but with a little practice you'll wonder how you ever coped with Basic's variables.

I can only begin to touch on some of the many features of Pascal in an article of this size, there hasn't been room to mention the "heap" and pointer variables, but it is worth mentioning some of the particular features of Lucidata Pascal. Programs are written as standard text files onto a Flex disc, then compiled into a "P-code" binary file. P-codes are opcodes for a theoretical computer which the runtime program interprets and executes. The result is an impressively fast program which is stored more compactly on disc than stand alone machine code. My only complaint is that the compiler does not recover well from program errors when compiling; once the initial error has been reported a large number of spurious error messages may apear before the compiler gets back on the "right track". There are plenty of standard methods for error recovery in compilers (such as Turner's Algorithm) and it's a pity that Lucidate do not seem to have used one.

Lucidata Pascal

Lucidata Pascal has a number of very useful string handling procedures predefined for the programmer (strings are traditionally the most weak area of Pascal) as well as some sophisicated file handling commands to access both sequential and random access Flex files. A potentially very useful feature is the "overlay" procedure which allows very large programs to be run by swapping in and out blocks of code during execution.

The accompanying manual gives a summary of all the features available and plenty of detail on non-standard additions to the language. There is also a large section on the internal implementation of the software to allow you to customise the run-time system by adding new built-in procedures (commands) etc. This section is certainly not for the novice and some knowledge of compiler construction is useful here.

In all important respects Lucidata Pascal conforms to the ISO Pascal Standard and most textbook programs will run without change. A number of demonstration programs are included on the disk, these are all rather elementary but do demonstrate some of the unique features of the implementation.

If this has whetted your appetite for more then there are plenty of books to be found on the Pascal Language. A few of the better ones to look out for are as follows:
PASCAL, An Introduction to Methodical Programming by Findlay and Watt.
PASCAL User Manual by Jensen and Wirth.
Programming in PASCAL by Peter Grogono.

System used: *LUCIDATA PASCAL* from Compusense — £90.
Requires: FLEX operating System (64K — Minimum 1 Disk Drive).

If you've got a technical question write to Brian Cadge. Please do not send a SAE as Brian cannot guarantee to answer individual inquiries.

Dragon Answers

THE CHANGE of style this month is not a new format — it's thanks to the Postie, who seems to have swallowed Brian Cadge's communique. Brian's faithful Dragon coughed up another copy of Dragon Answers, but not copies, alas, of the original questions. Brian has written a summary of the questions from memory — we hope you recognise your own problems!



Delta DOS

I HAVE a Dragon and Delta DOS and have transferred much of my software to disc. However, some games will not run if Delta is attached even though I can load them from disc and then relocate them. Is it possible to switch out Delta once the software has loaded from disc?

Andie Warner

IF YOU add the following lines of assembly language to your routine to relocate code it should allow most programs to run as if Delta Dos was not attached.

```
LDX  #40253
STX  269
LDX  #46185
STX  272
LDA  #57
STA  377
```

What this does is to reset the Interrupt vectors and also stops Delta from intercepting commands.

Interface

I'VE recently obtained a Tandy Coco disc drive and interface. Unfortunately, the interface does not seem to work with my Dragon. Is there a simple way of making it compatible, and if not where can I get a suitable controller?

Adrian Rothery

THE TANDY disc interface is certainly not compatible with the Dragon and there is no simple way to remedy this (you would have to change the ROM software to operate the cartridge with the Dragon).

However, Tandy disc drives are otherwise standard 5-inch drives and can be used with any suitable Dragon disc interface. Try contacting PNP Communications on 0273 514465, they can supply Dragondos compatible controllers for £70.

Extra RAM

I HAVE had a Dragon 64 for a couple of years and would like to know if it is possible to use the extra 32K of RAM to store graphics screens, when in 32K mode (with Dragondos).

Is this possible from Basic as I am a complete novice when it comes to machine code?

Gary Bell

TO USE the extra RAM for this purpose is not possible directly from Basic, however the program listed below will allow you to save and retrieve screens using the USR command from basic.

The routine automatically looks at the current graphics mode — where it starts and how much RAM is used and transfers this to the extra RAM at the offset given in the USR command. USR0 is used to store the screen, USR1 to retrieve it. There's room for 5 PMODE 3/4 screens or 10 PMODE 1/2 screens or a mixture of both. For example, to save a screen on display, at an offset of 6K in the extra RAM use X=USR0(6144), and then to retrieve it later use X=USR1(6144). The offset should be in the range 0 to 29400.

```
10 ,SAVE/LOAD GRAPHICS FOR
   64K
20 CLEAR 200,32700
30 FOR I=32701 TO
   32746:READ AS:POKE
   I,VAL("&H"+AS):NEXT
40 DEF USR0=32701:DEF
   USR1=32718
50 DATA 8D, 1C, A6, A0, A7, 80,
   10, 9C, B7, 25, F7, B7, FF,
   DE, 1C, EF, 39
60 DATA 8D, 0B, A, 80, A7, A0,
   10, 9C, B7, 25, F7, 20, ED,
   BD, 8B, 29, 8A
70 DATA 80, 1F, 01, 10, 9E, BA,
   1A, 10, B7, FF, DF, 39
```

Graphics

I HAVE been mucking about with the Dragon's semi-graphics modes. Could you please tell me how I can implement these modes while using my Cumana drive without corrupting the disk workspace.

J. H. Plester
47 Easington Road
Banbury
Oxon
OX16 9HJ

THE easiest way to set up the start address of the screen display is to use the normal Dragon Basic commands PMODE and SCREEN, followed by pokes to set up the semi-graphics mode. For example, to set up mode 24 starting at the first graphics page you could use the following lines:

```
10 PMODE 4,1:SCREEN 1,0
20 POKE &HFF22, (PEEK(&
   HFF22) AND 7)
```

New line

I HAVE recently obtained a printer for my Dragon computer, but when I list a program all I get is a one long string of output — how can I make the printer start a new line at the every listed line?

K. Moss

THIS question appears more regularly than most others (it's even more popular than the 'speed-up poke'), but is worth repeating occasionally for the benefit of new readers. The solution is to type the following immediately after powering up your Dragon:

POKE 155,80 (or 40 — number chars per line)
POKE 328,255:POKE 330,2

If this causes a blank line between each listed line then miss out the POKE 330,2. If this still doesn't work then POKE loc 331 with your printers' code for carriage return and loc 332 with the code for linefeed.

Routines

I AM trying to add some new commands to Basic and want to make use of some of the routines in the Delta DOS ROM. Can you tell me where I can obtain details of these routines?

K. McCandlish

TO MY knowledge the routine documentation for Delta/Cumana DOS has not been published and I have very little information on this DOS. Perhaps one of our readers has worked through the ROM and could help Mr McCandlish?

'Windows' solution

OVER a year after the 'Dragon Windows' program was published (July 1985) I am still getting letters about it so can I take this opportunity to answer all the queries in one go!

Due to a minor bug in the listing the program may occasionally crash on some machines. The solution is to add the following line to the Basic loader program:

6 POKE 31814,142

'Windows' cannot be used in 64K mode on a Dragon 64 as it uses a number of ROM calls, but will operate in 32K mode. Finally, many people have asked about using the software with Dragondos. There is no simple way of changing the loader to operate with Dragondos; a disk version has been written but this is a completely rewritten version and is not due to be published in *Dragon User*.

Screen Designer

***Dennis Riley** text, colour and graphics to create designer screens.*

THANKS TO ALL of the recent articles on machine coding, in particular the "Firmware" series, in *Dragon User*, I now find that programs in machine code are getting easier to write successfully. However, this left the problem of wanting programs to Auto-run, fortunately solved by Brian Cadge (Dragon Answers Aug '85), and to displaying a customised screen while loading. Hence Screen Designer.

Screen Designer allows a text screen to be built up using all of the graphics characters, colours and the majority of the text characters (some are used as function keys). A screen may be saved and re-loaded so that it may be re-used or modified. Text, colours and graphics characters can be edited and manipulated on screen to produce the desired effect. A program can be saved and made to Auto-run, displaying the screen created, and the program secured, in that pressing RESET will re-execute the program.

Entering the program

Two methods to enter the program are given. **Listing 1** gives a listing of the machine code which can be poked into memory, using the Basic loader provided (**listing 3**), which will allow you to enter the code in stages. Just enter as much as you like and CSAVE, then on re-starting CLOADM and continue from where you left off.

Listing 2 gives the Alldream assembler listing. It will be noticed from this listing that the program starts with a short piece of code which relocates the program to its workspace at loc 30001, this has been included as an example, the need for which will be given later.

When the program has been successfully entered save the program using:
CSAVEM "DESIGNER", &H0600, &H0CC1, &H0600
The program can now be run using:
CLOADM "DESIGNER": EXEC

Program operation

On running the program a Menu of options is given.

1) **Create Screen** This option is used to build up the required text screen and uses a number of function keys. On entering this option you will be faced with a green screen with a flashing cursor in the top left hand corner. Green is the default background colour (this is Graphics Green Code 143 and not text Green Code 32), this colour can be changed using the "$" key, by pressing the "$" key the colour will change for each press, and with the exception of the black screen a cursor will again appear, when text or a character is entered on to the screen or the cursor is moved.

The cursor can be postitioned anywhere on the screen, without destroying existing text or characters, by using the arrow keys. Auto-repeat is incorporated into the program to facilitate movement of the cursor and entering text or characters.

Text can be entered on to the screen as normal and deleted using the CLEAR key, it will be noticed that both the CLEAR key and SPACEBAR take on the background colour.

Graphics Characters are placed on the screen, under the cursor, using the "@" key; repeated pressing of this key will change the character shape.

The colour of the character under the cursor can be changed using Shift "@", again repeated pressing will change the colour, and the selected colour will remain in operation until altered.

Both the character shape and the colour are stored into memory, and can be repeated any where on the screen using the "+" key, here the auto-repeat makes light work of borders, though these are best left until last.

To help in the design of the screen there are six scrolling functions, the Shift arrow keys will scroll the whole screen in the appropriate direction and by using the ">" and "<" keys the individual line containing the cursor can be scrolled left or right.

Colour

The colour of all of the characters on screen can be altered using the "&" key, without affecting any text. However, as the colour of the space and delete will still be the original background colour, the use of this function is best left until a screen is completed.

The final function uses the "%" key and this will invert the character under the cursor, to obtain characters not normally available from the keyboard, i.e. inverse numerals.

The BREAK key will return to the Menu, and as the screen is saved to memory, obviously any previous screen will be lost.

2) **Display Screen** This option will display the screen held in memory, either one that is being worked on or one that has been loaded into memory. All of the described functions apply to this option.

3) **Save Screen** Here a screen can be saved to tape, and therefore a library of screens can be built up, or an unfinished screen can be saved for future work to be done on it.

4) **Load Screen** Loads a previously saved screen back into memory, together with the original background colour and the last character colour used, for use with the spacebar and delete.

Options 6 and 7 turn the motor on and off respectively, this is to allow for the cleaning and/or positioning of tapes. The BREAK key used from menu will return to basic.

Auto-running

Option 5 will convert a machine code program into one that will Auto-run and will display the screen chosen while loading. There are, however, some very important conditions which must be met for this to be successful.

The source program must have been saved using:
CSAVEM "TITLE", N1, N2, N3
Where:

N1 is the start address of &H0600. Programs must start at this address. Any program needing a higher workspace can be relocated using a similar method to the example given.

N2 is the end address of the program while in the graphics area.

Most importantly N3 is the address from which the source program is to be executed (EXED'd) when it is in its required workspace, even if this means that the EXEC address is higher than the end address. If N3 points to some other address it may result in the program crashing, and pressing reset, instead of re-executing the program will certainly cause it to crash. Ideally the EXEC address (N3) will be the same as the address JMP'd to in the relocation program. Of course if the program is to remain in the graphics area so much the better.

An example would be if Screen Designer itself were to be saved using Screen Designer.

Firstly the program would be loaded and executed, it is now in its workspace at loc 30001. However, as Screen Designer does not use Hires, the original program is still in the Graphics area, this can now be saved using:
CSAVEM "AUTODESN", &H0600, &H0CC1, &H7971
If the program is now saved, using Screen Designer, it will Auto-run on loading.

As can be seen, from the assembler listing, the relocation of code is very simple for anyone using the Alldream assembler, as there are two directives ORG and PUT which do the job for them.

Anyone referring to Brian Cadges piece on Auto-running machine code will see that he has given a little piece of code that should start any auto-run program, with Screen Designer. However, this is unnecessary as this code is already taken care of.

The first piece of code at the execution address should be a NOP, this is so that if the reset button is pressed the program will re-execute, if a NOP is not present, a cold start will occur.

Option 5 includes prompts where the motor is turned on or off to enable tapes to be correctly positioned.

Hints and tips

Most of the code used in this program was gleaned from the pages of Dragon User at some time or another, particularly in

regard to logic instructions. (Bruce Devlin D.U. Jan '84).

The routine called, using the "&" key, loads each character in turn from the text screen into the A accumulator; a text character is by-passed but a graphics character has its value increased by 16, to alter its colour. The A accumulator is then OR'd with #128 (&H80) to ensure that a graphics and not a text character results.

In Twos complement arithmetic numbers 0-127 represent positive numers (bit 7 is not set) and numbers 128-255 represent negative numbers (bit 7 is set), also, not by coincidence, numbers 0-127 represent text characters, whereas numbers 128-255 are graphics characters. Therefore, by using TST and the conditional branches BMI and, as in this case BPL, it is possible to distinguish between graphics and text characters, and Branch accordingly.

The previously described routine, if used as a subroutine, can do wonders for Menus or any text page that is waiting for a prompt, **Listing 4** gives a short example, a little over the top, maybe, but it does demonstrate what can be done to liven up a screen.

All of the I/O functions are from Brian cadge's "Firmware Series".

Entering machine code, especially without an assembler, can be laborious, so I will provide a copy of the program for £2 Write to Dennis Riley, 21 Colmore Road, Wortley, Leeds LS12 4DF. I will answer general questions about the program, but please send a stamped, self-addressed envelope.

Listing one: machine code data.

```
0600  8E7531108E0613ECA1ED = 1125
060A  818C7BDF26F77E797108 = 1268
0614  31033233343536377A2A = 531
061E  7A297A797A7E7AAE7B14 = 1093
0628  7ABD7AC31F2083753358 = 1078
0632  C3753A1F026EB4BD7A0C = 1016
063C  1F2083756958C3757A1F = 969
0646  026EB4120A5E09080340 = 498
0650  132B0C245F5B155D3C3E = 532
065A  2625779277A377C477B4 = 1236
0664  7A7077D477F678167827 = 1231
066E  78427869788C78B578E4 = 1320
0678  79187932794D79664900 = 810
0682  FF008F00000052455749 = 709
068C  4E4420534F5552434520 = 675
0696  5441504520544F205354 = 692
06A0  41525420414E44205052 = 668
06AA  4553532065 6E74657220 = 841
06B4  544F204C4F4144004C4F = 638
06BE  41442044455354494E41 = 685
06C8  54494F4E205441504520 = 676
06D2  414E4420202020202020 = 435
06DC  50524553532065 6E7465 = 857
06E6  7200504F534954494F4E = 743
06F0  2044455354494E415449 = 709
06FA  4F4E205441504520414E = 662
0704  44202020505245535320 = 593
070E  656E7465720053455420 = 810
0718  544F205245434F524420 = 674
0722  5448454E205052455353 = 732
072C  20656E7465722020544F = 801
0736  20534156452050524F47 = 679
0740  52414D004D454E550020 = 565
074A  20312E2E2E2E2E2E2E2E = 449
0754  2E2E2E2E2E2E43524541 = 559
075E  54452053435245454E0D = 646
0768  2020322E2E2E2E2E2E2E = 436
0772  2E2E2E2E2E2E44495350 = 580
077C  4C41592053435245454E = 710
0786  0D2020332E2E2E2E2E2E = 404
0790  2E2E2E2E2E2E2E2E2E2E = 460
079A  53415645205343524545 = 705
07A4  4E0D2020342E2E2E2E2E = 437
07AE  2E2E2E2E2E2E2E2E2E2E = 460
07B8  2E4C4F41442053435245 = 667
07C2  454E0D2020352E2E2E2E = 461
07CC  2E2E5341564520415554 = 661
07D6  4F2D52554E2050524F47 = 713
07E0  52414D0D2020362E2E2E = 493
07EA  2E2E2E2E2E2E2E2E2E2E = 460
07F4  2E2E2E2E2E2E4D4F544F = 595
07FE  52204F4E0D2020372E2E = 495
0808  2E2E2E2E2E2E2E2E2E2E = 460
0812  2E2E2E2E2E2E4D4F544F = 595
081C  52204F46460D20204252 = 558
0826  45414B2E2E2E2E2E2E2E = 531
0830  2E2E2E2E2E2E2E2E2E2E = 460
083A  2E2E515549540053454C = 643
0844  454354204E554D424552 = 709
084E  20524551554952454400 = 641
0858  66696C656E616D653A2D = 936
0862  00736176696E673A2D2D = 796
086C  FF00FF00FF00FF009E88 = 1314
0876  3088208C05FF1022029E = 826
0880  9F881602999E883088E0 = 1174
088A  8C04001025028D9F8816 = 657
0894  02889E88301F8C040010 = 671
089E  25027D9F881602789E88 = 897
08A8  30018C05FF1022026D9F = 769
08B2  88160268C6FF5CC40FCA = 1222
08BC  80F775A2FA7BD99E88E7 = 1769
08C6  84BD800627FB814027E8 = 1209
08D0  16025AC6F0F77BD9F67B = 1508
08DA  D9CB10C17022F2F77BD9 = 1604
08E4  FA75A29E88E784BD8006 = 1509
08EE  27FB81131026023720E0 = 805
08F8  9E88F675A2FA7BD9E784 = 1772
0902  30019F881602159E888C = 823
090C  04001027020CB67BD881 = 723
0916  7F26028B01A784A7829F = 1062
0920  881601FAC680F77BD8BD = 1510
092A  BA797A7BD8BD800627FB = 1381
0934  812410260 1E4F67BD8C1 = 1226
093E  FF24E3CB10F77BD8BDBA = 1698
0948  7920E4BD79B28E0400EC = 1251
0952  8820ED818C05E025F610 = 1202
095C  8E7BDA8E05E0ECA1ED81 = 1617
0966  8C060025F71601B0BD79 = 939
0970  B28E05FFA688E0A78430 = 1453
097A  1F8C041E22F4108E7BDA = 982
0984  31A901E08E0400ECA1ED = 1223
098E  818C042025F7160187BD = 936
0998  79B28E0400C61FA601A7 = 1008
09A2  805A26F930018C060025 = 737
09AC  F08E041F108E7BDAA6A4 = 1246
09B6  A78431A8203088208C06 = 910
09C0  0025F1160158BD79B28E = 1019
09CA  05FFC61FA61FA784301F = 1064
09D4  5A26F7301F8C040022EE = 870
09DE  8E0400108E7BDA31A81F = 893
09E8  A6A4A78431A820308820 = 1094
09F2  8C05E125F1160124BD7A = 1018
```

Listing one cont.

```
09FC  213088E0E6843404C61F = 1088
0A06  A601A7805A26F93504E7 = 1127
0A10  8416010ABD7A21301FE6 = 818
0A1A  843404C61FA61FA78430 = 961
0A24  1F5A26F73504E7841600 = 848
0A2E  EFBD79B28E0400A6844D = 1248
0A38  2A048B108A80A7808C06 = 908
0A42  0025F01600D69E88E684 = 1169
0A4C  C840E7841600CE120F6F = 999
0A56  7F75A5BDBA778E040E9F = 1222
0A60  888E7662BD7A158E0460 = 1068
0A6A  9F888E7667BD7A158E05 = 1137
0A74  E59F888E775FBD7A15BD = 1401
0A7E  800627FBF67531108E75 = 1111
0A88  32A1A01027FB9D5A26F7 = 1209
0A92  20E9BD7A0C8E0400108E = 892
0A9C  7BDAEC81EDA18C060025 = 1287
0AA6  F739108E04008E7BDAEC = 1185
0AB0  81EDA1108C060025F639 = 1029
0ABA  3426CCFFFFFD0150FD01 = 1392
0AC4  52FD0154FD0156FD0158 = 1102
0ACE  108E2328313F26FC35A6 = 854
0AD8  34127375A59E88A68488 = 1195
0AE2  40A7848E0472301F26FC = 992
0AEC  35927D75A52703BD79F6 = 1204
0AF6  393412A680BD800C4D26 = 865
0B00  F83592A6801F10C41F26 = 1053
0B0A  F839C68FF77BD8BDBA79 = 1728
0B14  8E04009F887301492003 = 665
0B1E  BD79D8BD79F6BD800627 = 1444
0B28  F8810D27F4108E7568F6 = 1298
0B32  7567A1A01027FAFF5A26 = 1229
0B3C  F7BD7A0C81202609B67B = 1083
0B46  D8817F26028B01BD800C = 981
0B50  20CC730149BD79B216FE = 1189
0B5A  FEBD79C620B4BD7AC9BD = 1675
0B64  BA778E04009F888E7781 = 1136
0B6E  BD7A158E797734108E7B = 1047
0B78  D8BF01E734101F10C302 = 951
0B82  0234068E7971BF01E534 = 909
0B8C  107E991BBD7AC9BDB8B3 = 1386
0B96  8E0000BDB74816FEBABD = 1237
0BA0  80157E799BBD80187E79 = 1139
0BAA  988608B701D1B775A3BD = 1342
0BB4  BA77CE77898E7776108E = 1304
0BBE  01D2BD7A15BD79F6BD80 = 1416
0BC8  0627F8810827DC810D27 = 870
0BD2  0EBD800CA7A0A7C07A75 = 1268
0BDC  A3270F20E28620A7A0A7 = 1135
0BE6  C07A75A326F72007BD79 = 1228
0BF0  F6810D26F939BDBA778E = 1369
0BFA  75A6BD7A15BD8015BD80 = 1270
0C04  06810D26F9BD8018BD7A = 1087
0C0E  C9BDB8B38E0000BDB748 = 1339
0C18  BF759EBE01E5BF75A0BD = 1543
0C22  BA778E75DABD7A15BD80 = 1431
0C2C  06810D26F9BD8015BDBA = 1148
0C36  778E7606BD7A15BD8006 = 1040
0C40  810D26F9BD8018BDBA77 = 1264
0C4A  8E7632BD7A15BD800681 = 1094
0C54  0D26F9BD7AC9BD79C68E = 1462
0C5E  0300108E7BC9A6A0A780 = 1106
0C68  4D26F9867EB701678E03 = 1056
0C72  00BF01688E7BB134108E = 948
0C7C  0167BF01E73410BE759E = 1060
0C86  3410BE75A0BF01E53410 = 1024
0C90  7E991B8639B701678EB4 = 1106
0C9A  4F9F728E0300C6104FA7 = 957
0CA4  805A26FB16FDAE8639B7 = 1330
0CAE  016732628E01E59F727E = 1071
0CB8  06000000000FF00FF00 = 516
```

Listing two: Alldream assembler listing.

```
0600  0600                ORG   $0600
0600 8E7531               LDX   #30001
0603 108E0613             LDY   #THERE
0607 ECA1        MOVELP   LDD   ,Y++
0609 ED81                 STD   ,X++
060B 8C7BDF               CMPX  #$7BDF
060E 26F7                 BNE   MOVELP
0610 7E7971               JMP   START
0613 12          THERE    NOP
7531  7531                ORG   30001
7531                      PUT   THERE
7531 08          MNTBNO   FCB   8
7532 3103323334  MENTAB   FCB   49,$03,50,51,52,53
7538 3637                 FCB   54,55
753A 7A2A        MNFNTB   FDB   CREATE
753C 7A29                 FDB   RETURN
753E 7A79                 FDB   DSPLAY
7540 7A7E                 FDB   SAVE
7542 7AAE                 FDB   LOAD
7544 7B14                 FDB   AUTO
7546 7ABD                 FDB   CASSON
7548 7AC3                 FDB   CASOFF
754A 1F20        MNCALC   TFR   Y,D
754C 837533               SUBD  #MENTAB+1
754F 58                   LSLB
7550 C3753A               ADDD  #MNFNTB
7553 1F02                 TFR   D,Y
7555 6EB4                 JMP   (,Y)
7557 BD7A0C      TABCHK   JSR   TESTFL
755A 1F20                 TFR   Y,D
755C 837569               SUBD  #TABLE+1
755F 58                   LSLB
7560 C3757A               ADDD  #TABSEL
7563 1F02                 TFR   D,Y
7565 6EB4                 JMP   (,Y)
7567 12          TABNUM   FCB   18
7568 0A5E090803  TABLE    FCB   $0A,$5E,$09,$08,$03
756D 4013280C24           FCB   $40,19,$28,12,$24
7572 5F5B155D3C           FCB   $5F,$5B,$15,$5D,$3C
7577 3E2625               FCB   $3E,$26,$25
757A 7792        TABSEL   FDB   DOWN
757C 77A3                 FDB   UP
757E 77C4                 FDB   RIGHT
7580 77B4                 FDB   LEFT
7582 7A70                 FDB   RTOMEN
7584 77D4                 FDB   CHASEL
7586 77F6                 FDB   COLSEL
7588 7816                 FDB   REPEAT
758A 7827                 FDB   DELETE
758C 7842                 FDB   BACKGD
758E 7869                 FDB   SCRLUP
7590 788C                 FDB   SCRLDN
7592 78B5                 FDB   SCRLLF
7594 78E4                 FDB   SCRLRT
7596 7918                 FDB   SCLNLF
7598 7932                 FDB   SCLNRT
759A 794D                 FDB   ALTCOL
759C 7966                 FDB   ORCHAR
759E             *STORE   RMB   2
75A0             EXECST   RMB   2
75A2 8F          CHABUF   FCB   143
75A3 00          COUNT    FCB   0
75A4             OFFSET   RMB   1
75A5 00          BLKFLG   FCB   0
75A6 524557494E  LDMESS   FCC   /REWIND SOURCE TAP/
75B7 4520544F20           FCC   /E TO START AND PR/
75C8 4553532065           FCC   /ESS enter TO LOAD/
75D9 00                   FCB   0
75DA 4C4F414420  SV3MES   FCC   /LOAD DESTINATION /
75EB 5441504520           FCC   /TAPE AND        PR/
75FC 4553532065           FCC   /ESS enter/,0
7606 504F534954  SV1MES   FCC   /POSITION DESTINAT/
7617 494F4E2054           FCC   /ION TAPE AND    PR/
7628 4553532065           FCC   /ESS enter/,0
7632 5345542054  SV2MES   FCC   /SET TO RECORD THE/
7643 4E20505245           FCC   /N PRESS enter  TO/
7654 2053415645           FCC   / SAVE PROGRAM/,0
7662 4D454E5500  MENMES   FCC   /MENU/,0
7667 2020        MN2MES   FCB   32,32
7669 312E2E2E2E           FCC   /1...............CR/
767A 4541544520           FCC   /EATE SCREEN/,13,32
```

Listing two cont.

```
7687 20                 FCB    32
7688 322E2E2E2E         FCC    /2...............DIS/
7699 504C415920         FCC    /PLAY SCREEN/,13,32
76A6 20                 FCB    32
76A7 332E2E2E2E         FCC    /3................../
76B8 5341564520         FCC    /SAVE SCREEN/,13,32
76C5 20                 FCB    32
76C6 342E2E2E2E         FCC    /4................../
76D7 4C4F414420         FCC    /LOAD SCREEN/,13,32
76E4 20                 FCB    32
76E5 352E2E2E2E         FCC    /5......SAVE AUTO-/
76F6 52554E2050         FCC    /RUN PROGRAM/,13,32
7703 20                 FCB    32
7704 362E2E2E2E         FCC    /6................../
7715 2E2E2E4D4F         FCC    /...MOTOR ON/,13,32
7722 20                 FCB    32
7723 372E2E2E2E         FCC    /7................../
7734 2E2E4D4F54         FCC    /..MOTOR OFF/,13,32
7741 20                 FCB    32
7742 425245414B         FCC    /BREAK............/
7753 2E2E2E2E2E         FCC    /.......QUIT/,0
775F 53454C4543  SELMES FCC    /SELECT NUMBER REQ/
7770 5549524544         FCC    /UIRED/,0
7776 66696C656E  FLNAME FCC    /filename:-/,0
7781 736176696E  SAVING FCC    /saving:-/
7789             NMBUFF RMB    8
7791 00                 FCB    0
7792 9E88        DOWN   LDX    $88
7794 308820             LEAX   32,X
7797 8C05FF             CMPX   #1535
779A 1022029E           LBHI   CRWT1
779E 9F88               STX    $88
77A0 160299             LBRA   CRWT1
77A3 9E88        UP     LDX    $88
77A5 3088E0             LEAX   -32,X
77A8 8C0400             CMPX   #1024
77AB 1025028D           LBLO   CRWT1
77AF 9F88               STX    $88
77B1 160288             LBRA   CRWT1
77B4 9E88        LEFT   LDX    $88
77B6 301F               LEAX   -1,X
77B8 8C0400             CMPX   #1024
77BB 1025027D           LBLO   CRWT1
77BF 9F88               STX    $88
77C1 160278             LBRA   CRWT1
77C4 9E88        RIGHT  LDX    $88
77C6 3001               LEAX   1,X
77C8 8C05FF             CMPX   #1535
77CB 1022026D           LBHI   CRWT1
77CF 9F88               STX    $88
77D1 160268             LBRA   CRWT1
77D4 C6FF        CHASEL LDB    #$FF
77D6 5C          CHWT01 INCB
77D7 C40F               ANDB   #$0F
77D9 CA80               ORB    #$80
77DB F775A2             STB    CHABUF
77DE FA7BD9             ORB    COLBUF
77E1 9E88               LDX    $88
77E3 E784               STB    ,X
77E5 BD8006      CHSLWT JSR    $8006
77E8 27FB               BEQ    CHSLWT
77EA 8140               CMPA   #$40
77EC 27E8               BEQ    CHWT01
77EE 16025A             LBRA   CRWT2
77F1 C6F0        RSTCOL LDB    #240
77F3 F77BD9             STB    COLBUF
77F6 F67BD9      COLSEL LDB    COLBUF
77F9 CB10               ADDB   #16
77FB C170               CMPB   #112
77FD 22F2               BHI    RSTCOL
77FF F77BD9             STB    COLBUF
7802 FA75A2             ORB    CHABUF
7805 9E88               LDX    $88
7807 E784               STB    ,X
7809 BD8006      CLSLWT JSR    $8006
780C 27FB               BEQ    CLSLWT
780E 8113               CMPA   #19
7810 10260237           LBNE   CRWT2
7814 20E0               BRA    COLSEL
7816 9E88        REPEAT LDX    $88
7818 F675A2             LDB    CHABUF
781B FA7BD9             ORB    COLBUF
781E E784               STB    ,X
7820 3001               LEAX   1,X
7822 9F88               STX    $88
7824 160215             LBRA   CRWT1
7827 9E88        DELETE LDX    $88
7829 8C0400             CMPX   #1024
782C 1027020C           LBEQ   CRWT1
7830 B67BD8             LDA    CHGSTR
7833 817F               CMPA   #127
7835 2602               BNE    NXTDEL
7837 8B01               ADDA   #1
7839 A784        NXTDEL STA    ,X
783B A782               STA    ,-X
783D 9F88               STX    $88
783F 1601FA             LBRA   CRWT1
7842 C680        BACKGD LDB    #128
7844 F77BD8             STB    CHGSTR
7847 BDBA79             JSR    $BA79
784A 7A7BD8             DEC    CHGSTR
784D BD8006      CHGWT1 JSR    $8006
7850 27FB               BEQ    CHGWT1
7852 8124               CMPA   #$24
7854 102601E4           LBNE   CRWT1
7858 F67BD8             LDB    CHGSTR
785B C1FF               CMPB   #255
785D 24E3               BHS    BACKGD
785F CB10               ADDB   #16
7861 F77BD8             STB    CHGSTR
7864 BDBA79             JSR    $BA79
7867 20E4               BRA    CHGWT1
7869 BD79B2      SCRLUP JSR    STRSCR
786C 8E0400             LDX    #1024
786F EC8820      SCUPL1 LDD    32,X
7872 ED81               STD    ,X++
7874 8C05E0             CMPX   #1504
7877 25F6               BLO    SCUPL1
7879 108E7BDA           LDY    #SCRNBF
787D 8E05E0             LDX    #1504
7880 ECA1        SCUPL2 LDD    ,Y++
7882 ED81               STD    ,X++
7884 8C0600             CMPX   #1536
7887 25F7               BLO    SCUPL2
7889 1601B0             LBRA   CRWT1
788C BD79B2      SCRLDN JSR    STRSCR
788F 8E05FF             LDX    #1535
7892 A688E0      SCDNL1 LDA    -32,X
7895 A784               STA    ,X
7897 301F               LEAX   -1,X
7899 8C041E             CMPX   #1054
789C 22F4               BHI    SCDNL1
789E 108E7BDA           LDY    #SCRNBF
78A2 31A901E0           LEAY   480,Y
78A6 8E0400             LDX    #1024
78A9 ECA1        SCDNL2 LDD    ,Y++
78AB ED81               STD    ,X++
78AD 8C0420             CMPX   #1056
78B0 25F7               BLO    SCDNL2
78B2 160187             LBRA   CRWT1
78B5 BD79B2      SCRLLF JSR    STRSCR
78B8 8E0400             LDX    #1024
78BB C61F        SCLFL2 LDB    #31
78BD A601        SCLFL1 LDA    1,X
78BF A780               STA    ,X+
78C1 5A                 DECB
78C2 26F9               BNE    SCLFL1
78C4 3001               LEAX   1,X
78C6 8C0600             CMPX   #1536
78C9 25F0               BLO    SCLFL2
78CB 8E041F             LDX    #1055
78CE 108E7BDA           LDY    #SCRNBF
78D2 A6A4        SCLFL3 LDA    ,Y
78D4 A784               STA    ,X
78D6 31A820             LEAY   32,Y
78D9 308820             LEAX   32,X
78DC 8C0600             CMPX   #1536
78DF 25F1               BLO    SCLFL3
78E1 160158             LBRA   CRWT1
78E4 BD79B2      SCRLRT JSR    STRSCR
78E7 8E05FF             LDX    #1535
78EA C61F        SCRTL2 LDB    #31
78EC A61F        SCRTL1 LDA    -1,X
78EE A784               STA    ,X
78F0 301F               LEAX   -1,X
78F2 5A                 DECB
78F3 26F7               BNE    SCRTL1
78F5 301F               LEAX   -1,X
78F7 8C0400             CMPX   #1024
78FA 22EE               BHI    SCRTL2
78FC 8E0400             LDX    #1024
78FF 108E7BDA           LDY    #SCRNBF
7903 31A81F             LEAY   31,Y
7906 A6A4        SCRTL3 LDA    ,Y
7908 A784               STA    ,X
790A 31A820             LEAY   32,Y
790D 308820             LEAX   32,X
7910 8C05E1             CMPX   #1505
7913 25F1               BLO    SCRTL3
7915 160124             LBRA   CRWT1
7918 BD7A21      SCLNLF JSR    CLCPOS
791B 3088E0             LEAX   -32,X
791E E684               LDB    ,X
7920 3404               PSHS   B
7922 C61F               LDB    #31
7924 A601        SLNLF1 LDA    1,X
7926 A780               STA    ,X+
7928 5A                 DECB
7929 26F9               BNE    SLNLF1
792B 3504               PULS   B
792D E784               STB    ,X
792F 16010A             LBRA   CRWT1
7932 BD7A21      SCLNRT JSR    CLCPOS
7935 301F               LEAX   -1,X
7937 E684               LDB    ,X
7939 3404               PSHS   B
793B C61F               LDB    #31
793D A61F        SLNRT1 LDA    -1,X
793F A784               STA    ,X
7941 301F               LEAX   -1,X
7943 5A                 DECB
```

Listing two cont.

```
7944 26F7                  BNE   SLNRT1
7946 3504                  PULS  B
7948 E784                  STB   ,X
794A 1600EF                LBRA  CRWT1
794D BD79B2        ALTCOL  JSR   STRSCR
7950 8E0400                LDX   #1024
7953 A684          ALTLP1  LDA   ,X
7955 4D                    TSTA
7956 2A04                  BPL   PRTCHR
7958 8B10                  ADDA  #$10
795A 8A80                  ORA   #$80
795C A780          PRTCHR  STA   ,X+
795E 8C0600                CMPX  #1536
7961 25F0                  BLO   ALTLP1
7963 1600D6                LBRA  CRWT1
7966 9E88          ORCHAR  LDX   $88
7968 E684                  LDB   ,X
796A C840                  EORB  #$40
796C E784                  STB   ,X
796E 1600CE                LBRA  CRWAIT
7971 12            START   NOP
7972 0F6F                  CLR   $6F
7974 7F75A5                CLR   BLKFLG
7977 BDBA77        MENPRT  JSR   $BA77
797A 8E040E                LDX   #1038
797D 9F88                  STX   $88
797F 8E7662                LDX   #MENMES
7982 BD7A15                JSR   SPRINT
7985 8E0460                LDX   #1120
7988 9F88                  STX   $88
798A 8E7667                LDX   #MN2MES
798D BD7A15                JSR   SPRINT
7990 8E05E5                LDX   #1509
7993 9F88                  STX   $88
7995 8E775F                LDX   #SELMES
7998 BD7A15                JSR   SPRINT
799B BD8006        SLWAIT  JSR   $8006
799E 27FB                  BEQ   SLWAIT
79A0 F67521                LDB   MNTBNO
79A3 108E7522              LDY   #MENTAB
79A7 A1A0          MNLOOP  CMPA  ,Y+
79A9 1027FB9D              LBEQ  MNCALC
79AD 5A                    DECB
79AE 26F7                  BNE   MNLOOP
79B0 20E9                  BRA   SLWAIT
79B2 BD7A0C        STRSCR  JSR   TESTFL
79B5 8E0400                LDX   #1024
79B8 108E7BDA              LDY   #SCRNBF
79BC EC81          STSCLP  LDD   ,X++
79BE EDA1                  STD   ,Y++
79C0 8C0600                CMPX  #1536
79C3 25F7                  BLO   STSCLP
79C5 39                    RTS
79C6 108E0400      DISSCR  LDY   #1024
79CA 8E7BDA                LDX   #SCRNBF
79CD EC81          DSSCLP  LDD   ,X++
79CF EDA1                  STD   ,Y++
79D1 108C0600              CMPY  #1536
79D5 25F6                  BLO   DSSCLP
79D7 39                    RTS
79D8 3426          REPKEY  PSHS  D,Y
79DA CCFFFF                LDD   #$FFFF
79DD FD0150                STD   $150
79E0 FD0152                STD   $152
79E3 FD0154                STD   $154
79E6 FD0156                STD   $156
79E9 FD0158                STD   $158
79EC 108E2328      DELAY1  LDY   #9000
79F0 313F          DLOOP1  LEAY  -1,Y
79F2 26FC                  BNE   DLOOP1
79F4 35A6                  PULS  D,Y,PC
79F6 3412          CBLINK  PSHS  A,X
79F8 7375A5                COM   BLKFLG
79FB 9E88                  LDX   $88
79FD A684                  LDA   ,X
79FF 8840                  EORA  #$40
7A01 A784                  STA   ,X
7A03 8E0473                LDX   #$0473
7A06 301F          BLNKLP  LEAX  -1,X
7A08 26FC                  BNE   BLNKLP
7A0A 3592                  PULS  A,X,PC
7A0C 7D75A5        TESTFL  TST   BLKFLG
7A0F 2703                  BEQ   TESTRT
7A11 BD79F6                JSR   CBLINK
7A14 39            TESTRT  RTS
7A15 3412          SPRINT  PSHS  A,X
7A17 A680          SPRLP1  LDA   ,X+
7A19 BD800C                JSR   $800C
7A1C 4D                    TSTA
7A1D 26F8                  BNE   SPRLP1
7A1F 3592                  PULS  A,X,PC
7A21 A680          CLCPOS  LDA   ,X+
7A23 1F10                  TFR   X,D
7A25 C41F                  ANDB  #$1F
7A27 26F8                  BNE   CLCPOS
7A29 39            RETURN  RTS
7A2A C68F          CREATE  LDB   #143
7A2C F77BD8                STB   CHGSTR
7A2F BDBA79                JSR   $BA79
7A32 8E0400        CREAT1  LDX   #1024
```

```
7A35 9F88                  STX   $88
7A37 730149                COM   329
7A3A 2003                  BRA   CRWAIT
7A3C BD79D8        CRWT1   JSR   REPKEY
7A3F BD79F6        CRWAIT  JSR   CBLINK
7A42 BD8006                JSR   $8006
7A45 27F8                  BEQ   CRWAIT
7A47 810D                  CMPA  #13
7A49 27F4                  BEQ   CRWAIT
7A4B 108E7568      CRWT2   LDY   #TABLE
7A4F F67567                LDB   TABNUM
7A52 A1A0          CHOLP1  CMPA  ,Y+
7A54 1027FAFF              LBEQ  TABCHK
7A58 5A                    DECB
7A59 26F7                  BNE   CHOLP1
7A5B BD7A0C                JSR   TESTFL
7A5E 8120                  CMPA  #32
7A60 2609                  BNE   NTSPAC
7A62 B67BD8                LDA   CHGSTR
7A65 817F                  CMPA  #127
7A67 2602                  BNE   NTSPAC
7A69 8B01                  ADDA  #1
7A6B BD800C        NTSPAC  JSR   $800C
7A6E 20CC                  BRA   CRWT1
7A70 730149        RTOMEN  COM   329
7A73 BD79B2                JSR   STRSCR
7A76 16FEFE                LBRA  MENPRT
7A79 BD79C6        DSPLAY  JSR   DISSCR
7A7C 20B4                  BRA   CREAT1
7A7E BD7AC9        SAVE    JSR   GETNAM
7A81 BDBA77                JSR   $BA77
7A84 8E0400                LDX   #$0400
7A87 9F88                  STX   $88
7A89 8E7781                LDX   #SAVING
7A8C BD7A15                JSR   SPRINT
7A8F 8E7977                LDX   #MENPRT
7A92 3410                  PSHS  X
7A94 8E7BD8                LDX   #CHGSTR
7A97 BF01E7                STX   $01E7
7A9A 3410                  PSHS  X
7A9C 1F10                  TFR   X,D
7A9E C30202                ADDD  #514
7AA1 3406                  PSHS  D
7AA3 8E7971                LDX   #START
7AA6 BF01E5                STX   $01E5
7AA9 3410                  PSHS  X
7AAB 7E991B                JMP   39195
7AAE BD7AC9        LOAD    JSR   GETNAM
7AB1 BD8BB3                JSR   $8BB3
7AB4 8E0000                LDX   #0
7AB7 BDB748                JSR   $B748
7ABA 16FEBA                LBRA  MENPRT
7ABD BD8015        CASSON  JSR   $8015
7AC0 7E799B                JMP   SLWAIT
7AC3 BD8018        CASOFF  JSR   $8018
7AC6 7E799B                JMP   SLWAIT
7AC9 8608          GETNAM  LDA   #$08
7ACB B701D1                STA   $01D1
7ACE B775A3                STA   COUNT
7AD1 BDBA77                JSR   $BA77
7AD4 CE7789                LDU   #NMBUFF
7AD7 8E7776                LDX   #FLNAME
7ADA 108E01D2              LDY   #$01D2
7ADE BD7A15                JSR   SPRINT
7AE1 BD79F6        FNWAIT  JSR   CBLINK
7AE4 BD8006                JSR   $8006
7AE7 27F8                  BEQ   FNWAIT
7AE9 8108                  CMPA  #$08
7AEB 27DC                  BEQ   GETNAM
7AED 810D                  CMPA  #$0D
7AEF 270E                  BEQ   CSFILL
7AF1 BD800C                JSR   $800C
7AF4 A7A0                  STA   ,Y+
7AF6 A7C0                  STA   ,U+
7AF8 7A75A3                DEC   COUNT
7AFB 270F                  BEQ   GTNMW2
7AFD 20E2                  BRA   FNWAIT
7AFF 8620          CSFILL  LDA   #$20
7B01 A7A0          FILLP   STA   ,Y+
7B03 A7C0                  STA   ,U+
7B05 7A75A3                DEC   COUNT
7B08 26F7                  BNE   FILLP
7B0A 2007                  BRA   GTNMRT
7B0C BD79F6        GTNMW2  JSR   CBLINK
7B0F 810D                  CMPA  #$0D
7B11 26F9                  BNE   GTNMW2
7B13 39            GTNMRT  RTS
7B14 BDBA77        AUTO    JSR   $BA77
7B17 8E75A6                LDX   #LDMESS
7B1A BD7A15                JSR   SPRINT
7B1D BD8015                JSR   $8015
7B20 BD8006        LOADWT  JSR   $8006
7B23 810D                  CMPA  #13
7B25 26F9                  BNE   LOADWT
7B27 BD8018                JSR   $8018
7B2A BD7AC9                JSR   GETNAM
7B2D BD8BB3                JSR   $8BB3
7B30 8E0000                LDX   #0
7B33 BDB748                JSR   $B748
7B36 BF759E                STX   XSTORE
7B39 BE01E5                LDX   $01E5
```

Listing two cont.

```
7B3C BF75A0                STX   EXECST
7B3F BDBA77                JSR   $BA77
7B42 8E75DA                LDX   #SV2MES
7B45 BD7A15                JSR   SPRINT
7B48 BD8006         SAVWT3 JSR   $8006
7B4B 810D                  CMPA  #13
7B4D 26F9                  BNE   SAVWT3
7B4F BD8015                JSR   $8015
7B52 BDBA77                JSR   $BA77
7B55 [illegible]           LDX   #SV1MES
7B58 BD7A15                JSR   SPRINT
7B5B BD8006         SAVWT1 JSR   $8006
7B5E 810D                  CMPA  #13
7B60 26F9                  BNE   SAVWT1
7B62 BD8012                JSR   $8012
7B65 BDBA77                JSR   $BA77
7B68 [illegible]           LDX   #SV3MES
7B6B BD7A15                JSR   SPRINT
7B6E BD8006         SAVWT2 JSR   $8006
7B71 810D                  CMPA  #13
7B73 26F9                  BNE   SAVWT2
7B75 BD7A19                JSR   GETNAM
7B78 [illegible]           JSR   DISSCR
7B7B 8E0300                LDX   #$0300
7B7E 108E7BC9              LDY   #SAVEDT
7B82 A6A0           SAVELP LDA   ,Y+
7B84 A780                  STA   ,X+
7B86 4D                    TSTA
7B87 26F9                  BNE   SAVELP
7B89 867E                  LDA   #$7E
7B8B B70167                STA   359
7B8E 8E0300                LDX   #$0300
7B91 BF0168                STX   360
7B94 8E7BB1                LDX   #SAVEND
7B97 3410                  PSHS  X
7B99 8E0167                LDX   #359
7B9C BF01E7                STX   $01E7
7B9F 3410                  PSHS  X
7BA1 BE759E                LDX   XSTORE
7BA4 3410                  PSHS  X
7BA6 BE75A0                LDX   EXECST
7BA9 BF01E5                STX   $01E5
7BAC 3410                  PSHS  X
7BAE 7E991B                JMP   39195
7BB1 8639           SAVEND LDA   #$39
7BB3 B70167                STA   359
7BB6 8EB44F                LDX   #$B44F
7BB9 9F72                  STX   $72
7BBB 8E0300                LDX   #$0300
7BBE C610                  LDB   #16
7BC0 4F                    CLRA
7BC1 A780           SVLOOP STA   ,X+
7BC3 5A                    DECB
7BC4 26FB                  BNE   SVLOOP
7BC6 16FDAE                LBRA  MENPRT
7BC9 8639B70167     SAVEDT FDB   $8639,$B701,$6722
7BCF 62BE01E59F            FDB   $62BE,$01E5,$9F72
7BD5 7E06                  FDB   $7E06
7BD7 00                    FCB   0
7BD9 00             CHGSTR FCB   0
7BD9 00             COLBUF FCB   0
7BDA                SCRNBF RMB   512
7DDA 12                    NOP
7DDB 12                    NOP
7DDC 12                    NOP
7DDD
7DDD
```

Listing three: Basic loader for machine code.

```
100 CLS:PRINT"ENTER START ADDRESS"
110 LINE INPUT"&H";AD$
120 AD=VAL("&H"+AD$)
130 CLS
140 PRINT
150 CK=0:CH=0:DT$="":A$=""
160 PRINT"LINE 0";HEX$(AD);":-"
170 PRINT
180 INPUT"ENTER CHECKSUM";CS
190 PRINT
200 PRINT@161,"DATA>";
210 FORK=1TO20
220 A$=INKEY$:IF (A$<"A" OR A$>"F") AND (A$<"0" OR A$>"9")  THEN220
230 DT$=DT$+A$:PRINT@166,;DT$;
240 NEXT
250 FORK=1TO20STEP2
260 CK=CK+VAL("&H"+MID$(DT$,K,2))
270 NEXT
280 IF CK<>CS THEN CLS:PRINT"DATA INCORRECT!! RE-ENTER":GOTO150
290 FORK=1TO20STEP2
300 POKEAD,VAL("&H"+MID$(DT$,K,2)):AD=AD+1
310 NEXT
320 IF AD>=&H0CC1 THEN CLS:PRINT"FINISHED":END
330 GOTO130
```

Listing four: shifting screen.

```
0600  0600                ORG    $0600
0600 8636         @START  LDA    #134
0602 8E0400               LDX    #1024
0605 A780         LOOP    STA    ,X+
0607 8B10                 ADDA   #16
0609 8A80                 ORA    #$80
060B 8C0600               CMPX   #1536
060E 25F5                 BCS    LOOP
0610 CC9F8F               LDD    #$9F8F
0613 8E0400               LDX    #1024
0616 E78820       EDGLP1  STB    32,X
0619 A780                 STA    ,X+
061B 8B10                 ADDA   #16
061D 8A80                 ORA    #$80
061F 8C0420               CMPX   #1056
0622 25F2                 BLO    EDGLP1
0624 CC8F9F               LDD    #$8F9F
0627 8E05C0               LDX    #1472
062A E78820       EDGLP2  STB    32,X
062D A780                 STA    ,X+
062F CB10                 ADDB   #16
0631 CA80                 ORB    #$80
0633 8C05E0               CMPX   #1504
0636 25F2                 BLO    EDGLP2
0638 CCAF8F               LDD    #$AF8F
063B 8E0420               LDX    #1056
063E ED84         EDGLP3  STD    ,X
0640 8B10                 ADDA   #16
0642 8A80                 ORA    #$80
0644 308820               LEAX   32,X
0647 8C05E0               CMPX   #1504
064A 25F2                 BLO    EDGLP3
064C CC8F9F               LDD    #$8F9F
064F 8E043E               LDX    #1086
0652 ED84         EDGLP4  STD    ,X
0654 CB10                 ADDB   #16
0656 CA80                 ORB    #$80
0658 308820               LEAX   32,X
065B 8C05E0               CMPX   #1504
065E 25F2                 BLO    EDGLP4
0660 8E044A               LDX    #1098
0663 9F88                 STX    $88
0665 8E06B6               LDX    #TEST
0668 BD06AA               JSR    SPRINT
066B 8E04EB               LDX    #1259
066E 9F88                 STX    $88
0670 8E06C2               LDX    #TEST2
0673 BD06AA               JSR    SPRINT
0676 8E05A8               LDX    #1448
0679 9F88                 STX    $88
067B 8E06CD               LDX    #TEST3
067E BD06AA               JSR    SPRINT
0681 BD068C       WAIT    JSR    ALTCOL
0684 BD8006               JSR    $8006
0687 8120                 CMPA   #32
0689 26F6                 BNE    WAIT
068B 39                   RTS
068C 3412         ALTCOL  PSHS   A,X
068E 8E0400               LDX    #1024
0691 A684         ALTLP1  LDA    ,X
0693 4D                   TSTA
0694 2A04                 BPL    PRTCHR
0696 8B10                 ADDA   #$10
0698 8A80                 ORA    #$80
069A A780         PRTCHR  STA    ,X+
069C 8C0600               CMPX   #1536
069F 25F0                 BLO    ALTLP1
06A1 8E4E20               LDX    #20000
06A4 301F         ALTLP2  LEAX   -1,X
06A6 26FC                 BNE    ALTLP2
06A8 3592                 PULS   A,X,PC
06AA 3412         SPRINT  PSHS   A,X
06AC A680         SPRLP1  LDA    ,X+
06AE BD800C               JSR    $800C
06B1 4D                   TSTA
06B2 26F8                 BNE    SPRLP1
06B4 3592                 PULS   A,X,PC
06B6 7465737480 TEST      FCC    /test/,128,/screen/
06C1 00                   FCB    0
06C2 6279806480 TEST2     FCC    /by/,128,/d/,128
06C7 7269606579           FCC    /riley/,0
06CD
06CD 7370616265 TEST3     FCC    /spacebar/,128,/to/
06D8 80656E6400           FCC    128,/end/,0
06DD
06DD
```

Sound Ability

The Dragon makes more noise than you think, says ***Jonathan Bates.***

I'M SURE that at one time or other you may have been jealous of the seemingly superior sound ability of other microcomputers such as the BBC or Atari machines. But the Dragon is equally able to create similar and even more advanced sound effects than these micros.

The problem is that the Dragon's sound facility is harder to use, as the other machines have specialist chips to handle their sound effects. But this is also a drawback for these machines as they are limited by the capabilities of their sound chips, whereas the Dragon's sound is only limited by your imagination.

The Dragon has excellent capabilities and is not limited by an inbuilt speaker linked directly to a sound chip like the system adopted for the BBC. This means we can relay sound from tape to the speaker (AUDIO ON) and record sound digitally in memory, a feature that is utilised in games such as *Tubeway Army and Dragrunner*. It is impossible to do this on the Beeb without adding additional hardware and so invalidating the warranty and risking damaging the machine.

Unfortunately the designers of the Dragon's basic interpreter did not utilise the sound capability to its full potential, and only gave us a SOUND and PLAY command, both of which are very limited and only give us one kind of note. I have written some short routines to demonstrate the potential of Dragon sound and although these routines are only the tip of the iceberg I think they will give you some inspiration to develop your own.

Analogue port

Listing 1 is a routine which uses the analogue port to free the sound generator. To use this, bit four of location &FF23 must be set. This is done by ORing it with 8. Then the routine takes a number from the low byte of the timer at location 275 and keeps decrementing the 'B' accumulator and storing the result in &FF20, the analogue port to which the sound generator is mapped, causing the effects. Only the top six bits of this port are used and the value placed in those bits corresponds to the volume of the click relayed to the loudspeaker. The rest of **listing 1** increments the accumulator, storing the result in &FF20 and, after doing this 255 times, the value in the accumulator returns to zero, setting the zero flag. This bypasses the Branch Not Equal instruction and Branches to the start again, repeating the process. This causes a pleasant sound effect which you can try yourself if you have an assembler, otherwise try typing in **listing 4**. I have located this routine at address 20000 but it is completely relocatable.

Even more effective is **listing 2** which I call 'Wangys'; if you type it in you will see and hear why. This routine uses the hires screen memory and simply operates an Exclusive OR on the accumulator from each memory location, leaves the result in that location and &FF20 and then moves on to the next location until it gets to the end of the screen. At this point it repeats itself. The Basic program, after setting up mode four, draws a circle, fills it in and then runs the routine. Starting off with different shapes or patterns causes completely different sound effects, so try replacing lines 20 to 90 with your own shapes. If you don't have an assembler, try **listing 5**.

Listing 3 extends the Dragon's inbuilt sound command and should be used in addition with the commands already used. It causes a note with different, buzzy harmonics and it can also be used very

quickly as it is in lines 60 to 90 in the accompanying Basic demonstration program.

The routine enables you to create a note of certain volume, pitch and length. The duration is controlled by the value in address ?4E60, the pitch is controlled by &4E62 and the volume by &4E64 although you can change these by changing their values at the end of the assembly listing, or as shown in the basic **listing 6**. The routine is also relocatable and it may be a good idea to shift it up in memory if you use it in a program where you are short on space.

The Dragon also has useful cassette port handling features and can be used for speech recognition, I have written a program to demonstrate this (**listing 7**). The program takes values from locations &FF20,&FF24,&FF28 and &FF30 and displays a quantity graph of the results on the screen. Each location gives a value of either one or zero depending on the sound from the cassette port. Try playing music from the cassette and watch the result, and if you find that any of the lines drift in a particular direction alter the volume and tone levels until they remain more or less straight. This level will then be your optimum volume and tone (if you have tone) levels for loading software. You will also see that different parts of the music make the some of the lines rise higher. Do not expect a sudden peak on a particularly loud note as the routine will simply make a line move up one pixel!

Listing one

```
4E2Ø  4E2Ø                  ORG   2ØØØØ
4E2Ø                        PUT   2ØØØØ
4E2Ø 86Ø8          @START   LDA   #8
4E22 BAFF23                 ORA   $FF23
4E25 B7FF23                 STA   $FF23
4E28 F6Ø113        LOOP1    LDB   275
4E2B F7FF2Ø        LOOP2    STB   $FF2Ø
4E2E 5A                     DECB
4E2F 26FA                   BNE   LOOP2
4E31 F7FF2Ø        LOOP3    STB   $FF2Ø
4E34 5C                     INCB
4E35 26FA                   BNEB  LOOP3
4E37 2ØEF                   BRA   LOOP1
4E39
```

Listing two

```
4E2Ø 86FF          @START LDA   #255
4E22 C6Ø8                 LDB   #8
4E24 FAFF23               ORB   $FF23
4E27 F7FF23               STB   $FF23
4E2A 8EØ6ØØ               LDX   #$6ØØ
4E2D B7FF2Ø        LOOP   STA   $FF2Ø
4E3Ø A884                 EORA  ,X
4E32 A78Ø                 STA   ,X+
4E34 8C1EØØ               CMPX  #$1EØØ
4E37 26F4                 BNE   LOOP
4E39 2ØE5                 BRA   @START
4E3B

1Ø PMODE4,1:SCREEN1,1:PCLS
15 REM *Try drawing different shapes e.t.c.
2Ø CIRCLE(128,96),5Ø
3Ø PAINT(128,96)
1ØØ EXEC 2ØØØØ
```

Listing three

```
4E2Ø B6FF23        @START LDA   $FF23
4E23 8AØ8                 ORA   #8
4E25 B7FF23               STA   $FF23
4E28 B64E6Ø               LDA   LENGTH
4E2B B74E61               STA   LEN
4E2E B64E62               LDA   PITCH
4E31 B74E63               STA   PIT
4E34 B64E64               LDA   VOLUME
4E37 49                   ROLA
4E38 49                   ROLA
4E39 C6ØØ                 LDB   #Ø
4E3B B7FF2Ø        SOUND  STA   $FF2Ø
4E3E F7FF2Ø               STB   $FF2Ø
4E41 7A4E63        LOOP1  DEC   PIT
4E44 26FB                 BNE   LOOP1
4E46 F64E62               LDB   PITCH
4E49 F74E63               STB   PIT
4E4C C6ØØ                 LDB   #Ø
4E4E B7FF2Ø               STA   $FF2Ø
4E51 F7FF2Ø               STB   $FF2Ø
4E54 7C4E63        LOOP2  INC   PIT
4E57 26FB                 BNE   LOOP2
4E59 7A4E61               DEC   LEN
4E5C 26DD                 BNE   SOUND
4E5E 39                   RTS

4E5F  4E6Ø         LENGTH EQU   $4E6Ø
4E5F  4E61         LEN    EQU   $4E61
4E5F  4E62         PITCH  EQU   $4E62
4E5F  4E63         PIT    EQU   $4E63
4E5F  4E64         VOLUME EQU   $4E64
4E5F

1Ø DURATION=&H4E6Ø:'location for length
2Ø PITCH=&H4E62:'location for pitch
3Ø VOLUME=&H4E64:'location for volume
4Ø POKEDURATION,1:'change duration
5Ø POKE VOLUME,255:'change volume
6Ø FOR A=1 TO 255
7Ø POKEPITCH,A:'change pitch
8Ø EXEC &H4E2Ø:'call routine
9Ø NEXTA
1ØØ POKEDURATION,RND(2Ø)
11Ø POKEVOLUME,RND(255)
12Ø POKEPITCH,RND(255)
13Ø EXEC&H4E2Ø
14Ø GOTO 1ØØ
15Ø 'volume,pitch and duration can be
    in the range (1 to 255)
```

Listing four

```
1Ø CLEAR2ØØ,19999:'This should always be x-1
2Ø X=2ØØØØ:'Change x to relocate
3Ø Y=X
4Ø READ A$
5Ø IF A$<>"END" THEN POKE X,VAL("&H"+A$):X=X+1:GOTO4Ø
6Ø EXEC Y
7Ø DATA86,Ø8,BA,FF,23,B7,FF,23,F6,Ø1,13,F7,FF,2Ø,5A,26,FA,F7,FF,2Ø,5C,26,FA,2Ø,E
F
8Ø DATA END
```

Listing five: Wangys

```
5 CLEAR200,19999:'This should always be x-1
10 PMODE4,1:SCREEN1,1:PCLS
15 GOSUB 1000
16 'Try drawing differents shapes e.t.c.
20 CIRCLE(128,96),50
30 PAINT(128,96)
100 EXEC Y
1000 X=20000
1010 Y=X
1020 READ A$
1030 IF A$<>"END" THEN POKEX,VAL("&H"+A$):X=X+1:GOTO1020
1040 RETURN
1050 DATA 86,FF,C6,08,FA,FF,23,F7,FF,23,8E,06,00,B7,FF,20,A8,84,A7,80,8C,15,00,2
6,F4,20,E5
1060 DATA END
```

Listing six

```
4 CLEAR200,19999:'This should always be x-1
5 GOSUB 1000:'The following pokes relocate the program
6 A$=HEX$(Y+64):POKEY+9,VAL("&H"+LEFT$(A$,2)):POKEY+10,VAL("&H"+RIGHT$(A$,2)):PO
KEY+58,VAL("&H"+LEFT$(A$,2)):POKEY+59,VAL("&H"+RIGHT$(A$,2))+1
7 POKEY+12,VAL("&H"+LEFT$(A$,2)):POKEY+13,VAL("&H"+RIGHT$(A$,2))+1:POKEY+34,VAL(
"&H"+LEFT$(A$,2)):POKEY+35,VAL("&H"+RIGHT$(A$,2))+3
8 POKEY+15,VAL("&H"+LEFT$(A$,2)):POKEY+16,VAL("&H"+RIGHT$(A$,2))+2:POKEY+39,VAL(
"&H"+LEFT$(A$,2)):POKEY+40,VAL("&H"+RIGHT$(A$,2))+2:POKEY+53,VAL("&H"+LEFT$(A$,2
)):POKEY+54,VAL("&H"+RIGHT$(A$,2))+3
9 POKEY+18,VAL("&H"+LEFT$(A$,2)):POKEY+19,VAL("&H"+RIGHT$(A$,2))+3:POKEY+21,VAL(
"&H"+LEFT$(A$,2)):POKEY+22,VAL("&H"+RIGHT$(A$,2))+4:POKEY+42,VAL("&H"+LEFT$(A$,2
)):POKEY+43,VAL("&H"+RIGHT$(A$,2))+3
10 DURATION=Y+64
20 PITCH=Y+66
30 VOLUME=Y+68
40 POKEDURATION,1
50 POKE VOLUME,255
60 FOR A=1 TO 255
70 POKEPITCH,A
80 EXEC Y
90 NEXTA
100 POKEDURATION,RND(20)
110 POKEVOLUME,RND(255)
120 POKEPITCH,RND(255)
130 EXEC Y
140 GOTO 100
1000 X=20000:'alter this value to where you want it relocating
1010 Y=X
1020 READA$
1030 IF A$<>"END" THENPOKEX,VAL("&H"+A$):X=X+1:GOTO1020
1040 RETURN
1050 DATAB6,FF,23,8A,08,B7,FF,23,B6,4E,60,B7,4E,61,B6,4E,62,B7,4E,63,B6,4E,64,49
,49
1060 DATAC6,00,B7,FF,20,F7,FF,20,7A,4E,63,26,FB,F6,4E,62,F7,4E,63,C6,00,B7,FF,20
1070 DATAF7,FF,20,7C,4E,63,26,FB,7A,4E,61,26,DD,39
1080 DATA END
```

Listing seven

```
10 CLS
20 INPUT"DO YOU WANT TO LISTEN TO INPUT";A$
30 IF LEFT$(A$,1)="Y" THEN AUDIOON ELSE AUDIOOFF
40 MOTORON
50 PMODE4,1:SCREEN1,1:PCLS
60 Y1=35
70 Y2=80
80 Y3=125
90 Y4=170
100 FOR A=0 TO 255
110 Y1=Y1+(PEEK(&HFF20)=1)-(PEEK(&HFF20)=0)
120 Y2=Y2+(PEEK(&HFF24)=1)-(PEEK(&HFF24)=0)
130 Y3=Y3+(PEEK(&HFF28)=1)-(PEEK(&HFF28)=0)
140 Y4=Y4+(PEEK(&HFF30)=1)-(PEEK(&HFF30)=0)
150 IF Y1<0 OR Y2<0 OR Y3<0 OR Y4<0 OR Y1>190 OR Y2>190 OR Y3>190 OR Y4>190 THEN
 50
160 PSET(A,Y1):PSET(A,Y2):PSET(A,Y3):PSET(A,Y4)
170 NEXTA
180 GOTO50
```

Sliding Graphics

***Pam D'Arcy** slips into something a little more BASIC.*

PERHAPS it is a sign of the maturity of the Dragon and its users that these days "Dragon User" features an abundance of machine code articles and, excellent though they may be, programs containing meaty hex dumps. However, I still come across many readers who are not interested in machine code and others who are put off by long listings, so once again I am offering a non-arcade game of minimum length, written entirely in BASIC, capable, of your own enhancements to suit. REM lines may be omitted for even faster completion. For those who like rather more than just a listing, some explanation of the Dragon BASIC graphics statements used completes the article.

Slide Puzzle Program (Listing 2)

The program takes the existing contents of graphics pages 1-4, copies them to pages 5-8 where it treats them as a 4★4 grid as in a sliding tile puzzle, scrambles the 'tiles', then waits for the player to reconstitute the original picture by moving the 'tiles' via the 'blank' square using the arrow keys. Movement is as per the plastic puzzle, that is, a 'tile' is moved into the 'blank' space. There is an abandon option (#) to allow the totally lost to start again!

Any existing PMODE3 screen may be used, although 'tile edging' and a contrasting coloured 'blank' square may need to be incorporated for easier operation. Instructions for doing this are detailed below. The 'default' position of the blank square is the top left of the screen. Meanwhile, a quick picture may be obtained from **Listing 1.**

A Picture Of A House (Listing 1)

I recommend that Line 540 is typed in first then to RUN the program after tying in each LINE/CIRCLE/PAINT statement which will show you the effect of each graphics statement and will confirm the validity of the statement as typed rather than laborious debugging after typing in the whole. When completed, delete line 540 if you intend to append **Listing 2** to form a single program.

An Existing Picture

Any existing PMODE3 graphics picture currently in graphics pages 1-4 may be used. You may have one available from a graphics generator program, light pen or touch pad creations, from interrupting (BREAK-RESET) a game containing a nice graphics display or one designed and drawn up by a program of your own making.

If you do not have a copy of the screen stored on tape or disk, I recommend that as the first step so that it can be reloaded another time, including if it gets messed up following these instructions! If needed, **Table 1** offers assistance with saving and loading the graphics screen.

Listing 3 (comprising an edited LINE 20, new LINE 30 and existing LINES 440-540 of **Listing 1**) is a program that will draw in the 'tile edging' and 'blank' the square in the top left position of the screen. To change the blank's colour, the first parameter of the COLOR statements should be changed to the number of the required colour. To change the position of the blank square, the appropriate co-ordinates as given in **Table 2** should be used in LINE 530. The default values for the position of the blank square in the Slide Program will need to be amended accordingly (see below).

LINE 30 makes a quick 'safety' copy of graphics pages 1-4 to pages 5-8 when RUN. Should you want to change something after the first RUN, a re-load of the original screen from tape or disk is unnecessary if you also edit LINE 30 to . . . PCOPY N+4 TO N . . . to restore the original copy when the program is subsequently RUN.

Having set up the tile edging and blank square, save the masterpiece to tape or disk for future loading to use with the Slide Program.

Program Techniques

Various 'default' values can quickly be changed to suit your requirements. These are centred on LINE 630:

CS=colour set (when RUN, key C changes colour set anyway)
DX, DY=default blank square co-ordinates of the 4★4 grid, 0-3 across (DX) by 0-3 down (DY), thus 0,0=top-left; 3,0=top right 0,3=bottom left; 3,3=bottom right, etc.
MV=value that is subject of RND to determine number of tile movements (when added to 6) to jumble the picture,
thus the initial default jumble is between 7 and 16 moves. (I use MV=16 for my children).

The GET statement copies a described rectangle from the screen display into an area of memory called an ARRAY VARIABLE. That Array Variable can then be copied to a different part of the screen using the PUT statement. In the Slide Program, we need to swap the blank square 'rectangle' with an adjacent 'rectangle' of a 'picture tile'. The contents of the blank square 'rectangle' never change so LINE 740 copies it to the array 'BS' where it stays for the duration of the run.

The important thing about the size of the array variable for GET/PUT is that the manual is wrong and greatly overstates the required size.

The width of the 'tiles' in pixel points for each of the four 'tiles' across the screen is 256/4=64 (variable PX). The depth of the 'tiles' in pixel points for each of the four 'tiles' down the screen is 192/4=48 (variable PY). According to the manual, this would require an array DIM BS(63,47) to store a copy of any of the 'tiles'. In fact, the graphics data is tightly packed into the array and a formula for calculating the required size that will work for all PMODES and GET/PUT options is:

TABLE 1
SAVE/LOAD FROM/TO GRAPHICS PAGES 1-4

System	SAVE	LOAD
Cassette only	CSAVEM"SCREEN", 1536, 1536+6144−1,6144	CLOADM"SCREEN"
DragonDOS 1.0	SAVE"SCREEN", 3072, 3072+6144, 6144	LOAD"SCREEN.BIN"
DragonDOS 4.0) & Cumana 2.0)	SAVE"SCREEN", 3072, 3072+6144−1, 6144	LOAD"SCREEN.BIN"
DeltaDOS	SAVEM"SCREEN", 1536, 1536+6144−1	LOADM"SCREEN"

TABLE 2
X, Y CO-ORDINATES TO FILL 'RECTANGLE' WITHIN 'EDGED TILE'

2,1	61,46	66,1	125,46	130,1	189,46	194,1	253,46
2,49	61,94	66,49	125,94	130,49	189,94	194,49	253,94
2,97	61,142	66,97	125,142	130,97	189,142	194,97	253,142
2,145	61,190	66,145	125,190	130,145	189,190	194,145	253,190

PX=Width of rectangle involved in PIXEL POINTS
PY=depth of rectangle involved in PIXEL POINTS
Then ARRAY VARIABLE size=INT ((INT (((PX★PY)+7)/8)+4)/5)

Thus, typing into the Dragon the above information in COMMAND MODE

PX=64 <ENTER>
PY=48 <ENTER>
PRINT INT((INT(((PX★PY)+7)/8)+4)/5) <ENTER>

reveals a required DIM Array size of 77 (so even the BS(80) slightly exceeds the minimum requirement!).

There is no harm in overstating the required size, apart from wasting memory, but there is every harm in understating the size. Also, unlike the normal use of arrays in BASIC where if an array that has not been specifically DIMensioned is encountered, BASIC automatically allocates a 'default' size of 10, the array MUST be defined in a DIM statement as BASIC will flag an ERROR and NOT default define the ARRAY in these circumstances. The purpose of the LD variable checked in line 820 is to avoid the next tile move when jumbling the picture to simply mirror the previous move, thus negating some of the effect of the moves.

The GET/PUT subroutine in lines 1040-1080 swaps the tile to be moved with the adjacent 'blank' square. At this point, the 4★4 grid co-ordinates are held in MX and MY for the 'picture tile' being moved and BX and BY for the current position of the 'blank square'. First, the 'picture tile rectangle' is GET into the MS Array variable. The 'blank square rectangle' copied into array variable BS at the start of the run is PUT over that tile on the screen. The 'picture tile rectangle' in array MS is then PUT over the 'blank square rectangle' that it is being moved to. the blank square co-ordinates are then updated to its new position (line 1070).

Avoiding FC Errors

If the Slide Program is abandoned by, say, pressing BREAK rather than using the '★' option, and the next PCLEAR statement used is other than PCLEAR8, either directly from the keyboard or even in a newly loaded program prior to a PMODE statement being issued, an FC ERROR will occur.

This is because the Slide program was currently set to PMODE 3,5 thus using graphics pages 5-8 and the BASIC interpreter safeguards the graphics pages needed by the current PMODE setting. A lower PCLEAR figure will not be allowed until the machine has had its PMODE setting suitably reset. Thus when the program is quit through its in-built option (★), the PMODE setting is set to the lowest option requiring just one graphics page, PMODE0,1 prior to the END statement (line 950). The PCLEAR8 is reset to PCLEAR4 to release the unneeded 'working' pages of the slide puzzle, but retaining the 'master' pages 1-4 intact in case the program is re-RUN. A BASIC program is physically moved down into the freed graphics pages as soon as a lower PCLEAR statement is issued; try quitting the program using the '★' option, typing in PCLEAR1<ENTER> from the keyboard then RUN<ENTER>. The visible graphics screen corruption is where the Slide Program was moved to following PCLEAR1. It is physically copied to higher in memory when a greater number of graphics pages than at present are reserved with PCLEAR. It is quite a good idea to commence graphics programs with a PMODE0,1 so that FC Errors are avoided when setting up graphics requirements regardless of the state that a previous program may have left the machine in.

Picture Programming

I have spent rather longer than intended on the Slide program so will just mention one or two brief points regarding the Picture Program. In PMODE3, the size of the colour unit is 2 pixel points wide by 1 pixel point deep. The co-ordinates used are still based on the highest resolution graphics screen, being 256 pixel points across (and 192 points down). They are addressed as 0-255 and 0-191 respectively. If an odd pixel co-ordinate is specified when referencing a point across the screen (= X co-ordinate), that co-ordinate has 1 subtracted from it when colouring the screen. That is, as in the program where an X co-ordinate such as 255 is used (line 60), the blob of colour painted in occupies the two pixel points 254 and 255. The size of the colour unit is why a STEP of 2 is used in line 80.

LINE statements with no concluding B or BF parameter result in a line being drawn from the first pair of X, Y co-ordinates to the second pair. LINE statements concluded with a B alone means 'draw a rectangle (Box) in outline only', the pairs of X, Y co-ordinates defining the diagonally opposite corners of the rectangle. The BF parameter on the LINE statements means 'draw and Fill this rectangle (Box) with the current foreground colour'. Only rectangles can be automatically filled with colour. Other shapes, such as the circle of sunshine and the roof of the house, need to be PAINTed after drawing the outline. The area to be PAINTed needs to be completely outlined with a defined single colour border otherwise the 'paint' will spread alarmingly!

Listing 1

```
10 REM DRAW PICTURE OF A HOUSE FOR SLIDE PUZZLE PROGRAM
20 PCLEAR4:PMODE3,1:COLOR3,1:SCREEN1,0
30 REM GREEN GRASS
40 PCLS
50 REM BLUE SKY
60 LINE(0,0)-(255,103),PSET,BF
70 REM MAKE HORIZON LIGHTLY JAGGED AS PER GRASS GROWING
80 FOR X=0 TO 254 STEP 2
90 N=RND(2):IF N=1 THEN PSET(X,104)
100 NEXT X
110 REM YELLOW HOUSE BLOCK
120 COLOR2,1:LINE(60,64)-(199,179),PSET,BF
130 REM ROOF OUTLINE
140 COLOR4,1
150 LINE(80,32)-(180,32),PSET
160 LINE(180,32)-(202,63),PSET
170 LINE(202,64)-(56,64),PSET
180 LINE(56,63)-(80,32),PSET
190 REM PAINT ROOF RED
200 PAINT(126,48),4,4
210 REM RED CHIMNEY
220 LINE(120,20)-(135,32),PSET,BF
230 REM BLUE DOOR
240 COLOR3,1
250 LINE(112,120)-(147,179),PSET,BF
260 REM FOUR RED CURTAINED WINDOWS
270 COLOR4,1
280 LINE(68,76)-(101,107),PSET,BF
290 LINE(156,76)-(189,107),PSET,BF
300 LINE(68,130)-(101,161),PSET,BF
310 LINE(156,130)-(189,161),PSET,BF
320 REM OPEN THE CURTAINS
330 COLOR3,1
340 LINE(78,76)-(91,107),PSET,BF
350 LINE(166,76)-(179,107),PSET,BF
360 LINE(78,130)-(91,161),PSET,BF
370 LINE(166,130)-(179,161),PSET,BF
380 REM YELLOW OUTLINED DOOR KNOB
390 CIRCLE(138,156),4,2
400 REM YELLOW SUNSHINE
410 CIRCLE(216,22),10,2
420 COLOR2,1
430 PAINT(216,22),2,2
440 REM EDGE TILES IN RED
450 COLOR4,1
460 FOR X=0 TO 192 STEP 64
470 FOR Y=0 TO 144 STEP 48
480 LINE(X,Y)-(X+63,Y+47),PSET,B
490 NEXT Y
500 NEXT X
510 REM SET UP BLANK TILE IN YELLOW, TOP LEFT
520 COLOR2,1
530 LINE(2,1)-(61,46),PSET,BF
540 IF INKEY$="" THEN540 ELSE LIST
```

Listing 2

```
600 REM SLIDE PUZZLE - PAM D'ARCY - MAY 1986
610 PCLEAR8
620 DIM BS(80),MS(80)
630 CS=0:PX=64:PY=48:DX=0:DY=0:AU$=CHR$(94):AD$=CHR$(10):AL$=CHR$(8):AR$=CHR$(9)
:MV=10
640 CLS:PRINT"SLIDE PUZZLE"
650 PRINT:PRINT"USE ARROW KEYS TO MOVE TILES"
660 PRINT:PRINT"OTHER KEYS ARE:":PRINT"# TO RESTART; * TO QUIT;"
670 PRINT"C TO CHANGE COLOUR SET"
680 PRINT:PRINT"PRESS A KEY TO START"
690 M=RND(MV):K$=INKEY$:IF K$=""THEN690
700 BX=DX:BY=DY
710 PMODE3,1
720 REM COPY BLANK TILE TO ARRAY
730 REM SLIDING TAKES PLACE IN PAGES 5-8
740 GET(BX*PX,BY*PY)-((BX*PX)+PX-1,(BY*PY)+PY-1),BS,G
750 PMODE3,5:SCREEN1,CS
760 FOR N=1 TO 4:PCOPY N TO N+4:NEXT N
770 REM NUMBER OF MOVES TO MIX UP PUZZLE
780 M=RND(MV)+6
790 LD=0:BX=DX:BY=DY
800 FOR N=1 TO M
810 MX=BX:MY=BY
820 D=RND(4):IF D=LD THEN820
830 ON D GOTO840,850,860,870
840 MY=BY+1:LW=2:GOTO880
850 MY=BY-1:LW=1:GOTO880
860 MX=BX+1:LW=4:GOTO880
870 MX=BX-1:LW=3
880 IF MX<0 OR MX>3 OR MY<0 OR MY>3 THEN 810
890 GOSUB1040
900 LD=LW:NEXT N
910 REM READY TO BE UNSCRAMBLED
920 K$=INKEY$:IF K$=""THEN920
930 IF K$="C" THEN CS=1-CS:SCREEN1,CS:GOTO920
940 IF K$="#" THEN760
950 IF K$="*" THEN PMODE0,1:PCLEAR4:END
960 MX=BX:MY=BY
970 IF K$=AU$ THEN MY=BY+1:GOTO1010
980 IF K$=AD$ THEN MY=BY-1:GOTO1010
990 IF K$=AL$ THEN MX=BX+1:GOTO1010
1000 IF K$<>AR$ THEN 920 ELSE MX=BX-1
1010 IF MX<0 OR MX>3 OR MY<0 OR MY>3 THEN 920
1020 GOSUB1040
1030 GOTO920
1040 GET(MX*PX,MY*PY)-((MX*PX)+PX-1,(MY*PY)+PY-1),MS,G
1050 PUT(MX*PX,MY*PY)-((MX*PX)+PX-1,(MY*PY)+PY-1),BS,PSET
1060 PUT(BX*PX,BY*PY)-((BX*PX)+PX-1,(BY*PY)+PY-1),MS,PSET
1070 BX=MX:BY=MY
1080 RETURN
```

Listing 3

```
20 PCLEAR8:PMODE3,1:COLOR3,1:SCREEN1,0
30 FOR N=1 TO 4:PCOPY N TO N+4:NEXT
440 REM EDGE TILES IN RED
450 COLOR4,1
460 FOR X=0 TO 192 STEP 64
470 FOR Y=0 TO 144 STEP 48
480 LINE(X,Y)-(X+63,Y+47),PSET,B
490 NEXT Y
500 NEXT X
510 REM SET UP BLANK TILE IN YELLOW, TOP LEFT
520 COLOR2,1
530 LINE(2,1)-(61,46),PSET,BF
540 IF INKEY$="" THEN540 ELSE LIST
```

Classified

Expert's Arcade Arena

Write to 'The Expert' at Dragon User 12-13 Little Newport St, London WC2H 7PP, with all your arcade tips and hints.

GOOD DAY to you all and welcome to the fourth arcade column and once again may I thank you for all the letters you have sent. I have bought a warehouse to store them both in so fear not for their safety. However, should you ever wish to see anything you send me again please send an SAE or alternatively leave ten thousand pounds in a hollow tree in Dagenham by Monday or I'll start sending them back. In bits. Heh, heh, heh.

Firstly the winner of a year's subscription is a Mr David Barclay of Dumfries, Scotland (you know, the pointy bit up the top with the haggis and the lochs) who correctly named pictures (a) and (b) as *3-D Seiddab Attack* and *Horace Goes Skiing* respectively. Congratulations David and tough (censored) to the rest of you, especially those of you who can't spell *Seiddab*.

Now then, moving at the speed of light on to *Total Eclipse* (which isn't strictly an arcade game but still seems to be the subject of several thousand letters to me) and two different ways of helping yourselves along with this game.

Firstly, you write to Eclipse-Fenmar who expressed an interest up at the recent 6809 convention in selling saved games for a couple of quid at various points in the game. This sounds to me like a pretty groovy idea but it's up to you to nag them!

If you don't feel like doing that then there is another solution. Over to J. Brown of Buckinghamshire with his "Total Eclipse Savegame Editor". Mr Brown's program will only run on a Dragon 64 and as I only possess a Dragon 32 (unless any company out there really feels benevolent and wants to butter up a writer!) I have not tested it.

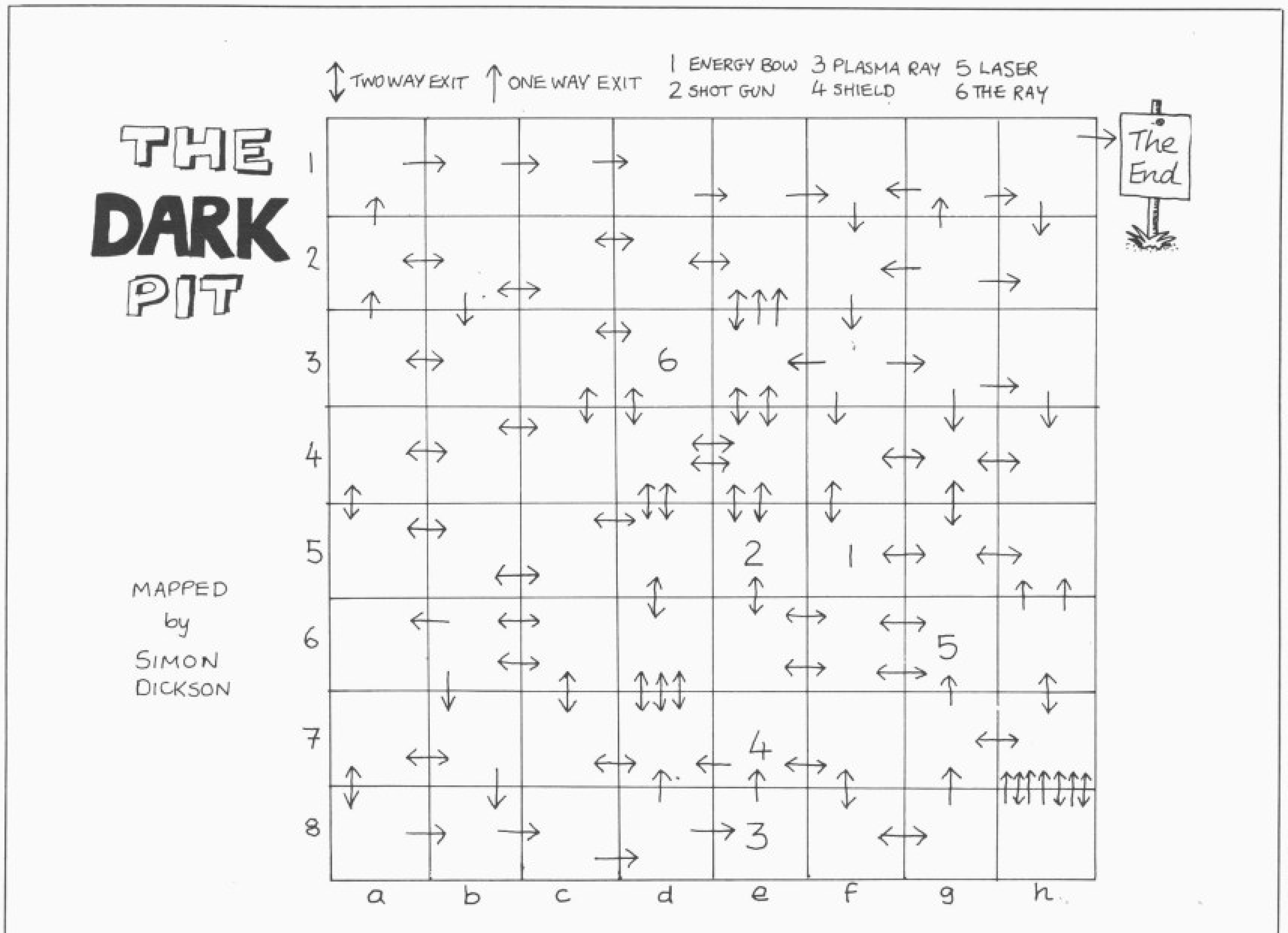
Mr Brown has sent complex instructions but the program is very easy to operate and they're not really necessary. Here are the prominent excerpts:

"The program only works on a D64 in 64k mode but can be typed in on a 32. Before loading however the 64k mode must have been selected . . . For use with option 8, locations 32115 to 32122 for investing, 32134 to 32136 will change the number of shards, disks, and pills you are carrying . . . Within option 9 'Q' returns you to the memory and 'S' will hold the listing or when held down make the listing slower . . . The maximum number of credits is 4290067295. Any more and the program will crash!"

Thank you very much, Mr Brown; some more *Total Eclipse* next month, now to *The Dark Pit* and the map from Simon Dickson. To be brutally honest I think that there's an error on this map as looking at it there seems to be no way to get to the exit, but my copy is mangled and so until my new one arrives I can't test it. However, rather than hold it back from you I've decided to go public so here's the map and please, don't send me any more maps of this, or of *Jet Set Willy* as although the thought is very nice, they're no longer necessary! However, do keep sending maps of any game you have (*Brocks Kingdom,* anyone?), I have even given consideration to publishing maps of adventure games (look out Mike Gerrard, this is a takeover bid) but please, please, please, if you want them returned make sure your name and address is on the map as well as the letter you send with it! Better still, keep a photocopy.

Moving on, an appeal now from Jonathan Harrap of Oxon: "In *Time Bandit* I have successfully completed 4D on 'The Time Gate' and have been returned to 1A, but I failed to do this on any of the other screens so I have not found out what the secret message is. I am now completely bored and thoroughly sick with this game. Can you please tell me what the message is?" Answers on a postcard to the usual address!

And so to a man who is becoming a co-writer on this column!! Mr M. R. Vince who reliably informs me that his name is "Mick"/"Mike the Brave" or "Hey You" . . . doesn't explain the 'R' Mick! Still, Mick tells us that the access code for *Beanstalker* is



```
FORI=1TO12:READA$(I):NEXT:FORI=1TO7:READB$(I),BM(I):NEXT
20 REM***************************
30 REM******total eclipse******
40 REM******     editor    ******
50 REM******  by j brown ******
60 REM******(C) superdata******
70 REM***************************
80 Z$=CHR$(128):CLS0:PRINT@6,"total";Z$;"eclipse";Z$;"editor";
90 PRINT@75,"by";Z$;"j";Z$;"brown";
100 PRINT@129,"menu";
110 PRINT@160,"1) POSITION"
120 PRINT@192,"2) CASH"
130 PRINT@224,"3) EQUIPMENT"
140 PRINT@256,"4) CARGO"
150 PRINT@288,"5) QUIT"
160 PRINT@320,"6) LOAD"
170 PRINT@352,"7) SAVE"
180 PRINT@384,"8) EDIT BYTE"
190 PRINT@416,"9) LIST MEMORY"
200 SCREEN0,1
210 A$=INKEY$:IFA$=""THEN210
220 PRINT@480,"OPTION ";A$;
230 IF ASC(A$)<49 OR ASC(A$)>57 THEN 200
240 ON ASC(A$)-48 GOTO 250,410,500,740,1390,850,970,1140,1230
250 REM POSITION
260 CLS
270 PRINT@0,"X CO-ORD. ",PEEK(32266)
280 PRINT@32,"Y CO-ORD. ",PEEK(32267)
290 PRINT@64,"SECTOR. ",PEEK(32304)
300 PRINT@96,"GALAXY. ",PEEK(32307)
310 PRINT@160,"";:INPUT"X";X:IFX<1ORX>6THEN310
320 POKE32266,X
330 PRINT@160,"";:INPUT"Y";Y:IFY<1ORY>6THEN330
340 POKE32267,Y
350 PRINT@160,"";:INPUT"SECTOR";S:IFS<1ORS>10THEN350
360 POKE32304,S
370 PRINT@160,"";:INPUT"GALAXY";G:IFG<1ORG>12THEN370
380 POKE32307,G
390 INPUT"GOOD Y-1,N-0";AN
400 IF -Y THEN80 ELSE 260
410 REM CASH
420 CLS:PRINT USING"%   %###,###,###,###";"CASH ",PEEK(32053)*16777216+PEEK(3205
4)*65536+PEEK(32055)*256+PEEK(32056)
430 PRINT@64,"";:INPUT"CASH";CS:IF CS<0 OR CS>18955840491THEN430
440 C1=INT(CS/16777216):CS=CS-C1*16777216
450 C2=INT(CS/65536):CS=CS-C2*65536
460 C3=INT(CS/256):CS=CS-C3*256
470 POKE32053,C1:POKE32054,C2:POKE32055,C3:POKE32056,CS
480 INPUT"GOOD Y-1,N-0";AS
490 IF -AS THEN 80 ELSE 430
500 REM EQUIPMENT
510 GOSUB 630
520 PRINT@256,"ITEM (INITIAL)";:INPUT AS$
530 FORI=1TO7
540 IF AS$=LEFT$(B$(I),1)THENIT=I:I=10
550 NEXT
560 IF I=8 THEN520
570 PRINT@256,"QUANTITY MAX(";BM(IT);")";:INPUTQT
580 IF QT<0 OR QT>BM(IT) THEN 570
590 POKE32085+IT,QT
600 GOSUB 630
610 INPUT"GOOD 1-Y,0-N";AS
620 IF -AS THEN 80 ELSE 510
630 CLS:LC=32086
640 FORI=1TO7
650 PRINTB$(I);BM(I),PEEK(LC)
660 LC=LC+1
670 NEXT:RETURN
680 REM CARGO
690 CLS:LC=32093
700 FORI=1TO12
710 PRINTA$(I),PEEK(LC)
720 LC=LC+1:NEXT
730 RETURN
740 GOSUB680:PRINT@416,"ITEM (INITIAL)";:INPUTAS$
750 FORI=1TO12
760 IF AS$=LEFT$(A$(I),1) THEN IT=I:I=15
770 NEXT
780 IF I=13 THEN 740
790 PRINT@416,"QUANTITY MAX(255)";:INPUTQT
800 IF QT<0 OR QT>255 THEN 790
810 POKE32092+IT,QT
820 GOSUB 680
830 PRINT@416,"GOOD 1-Y,0-N";:INPUTASAS
840 IF -AS THEN 80 ELSE 740
850 REM LOAD
860 CLS0:PRINT@0,"save";Z$;"game";Z$;"loader";:PRINT@32,"motor";Z$;"on";
870 MOTORON:AUDIOON
880 PRINT@66,"Position";Z$;"tape";
890 PRINT@128,"Press";Z$;"any";Z$;"key";Z$;"when";Z$;"ready";
900 A$=INKEY$:IFA$=""THEN900
910 MOTOROFF
920 PRINT@192,"Press";Z$;"play";Z$;"then";Z$;"any";Z$;"key";
930 A$=INKEY$:IFA$=""THEN930
940 PRINT@256,"loading";
950 CLOADM
960 GOTO50
970 REM SAVE
980 CLS0:PRINT@0,"save";Z$;"game";
990 MOTORON:AUDIOON
1000 PRINT@32,"motor";Z$;"on";
1010 PRINT@66,"Position";Z$;"tape";
1020 PRINT@120,"Press";Z$;"any";Z$;"key";Z$;"when";Z$;"ready";
1030 A$=INKEY$:IFA$=""THEN1030
1040 MOTOROFF
1050 PRINT@192,"PRESS PLAY AND RECORD THEN PRESSANY KEY";
1060 A$=INKEY$:IFA$=""THEN1060
1070 PRINT@256,"saving";
1080 CSAVEM"CDATA000",31919,32767,0
1090 PRINT@320,"reposition";Z$;"tape";
1100 MOTORON
1110 A$=INKEY$:IFA$=""THEN1110
1120 MOTOROFF
1130 GOTO 50
1140 REM BYTE EDITOR**
1150 CLS:INPUT"LOCATION(31919 TO 32767)";LC
1160 IF LC<31919 OR LC>32767 THEN 1150
1170 PRINT@32,"PRESENT DATA IS";PEEK(LC)
1180 PRINT@64,"NEW DATA";:INPUTND
1190 IF ND<0 OR ND>255 THEN 1180
1200 PRINT@96,"NEW DATA AT";LC;"IS";PEEK(LC)
1210 PRINT@128,"GOOD 1-Y,0-N";:INPUTAS
1220 IF -AS THEN 50 ELSE 1140
1230 REM MEMORY LIST
1240 BT=31919:HE=32767
1250 CLS:INPUT"START";ST
1260 IF ST<31919 OR ST>32766 THEN 1250
1270 PRINT@32,"END";:INPUTED
1280 IF ED<ST OR ED>32767 THEN1270
1290 FORI=ST TO ED
1300 P1=PEEK(I):P2=PEEK(I)*256+PEEK(I+1):P3=PEEK(I)*65536+PEEK(I+1)*256+PEEK(I+2
)
1310 PRINTUSING"##### ### #####  ########";I,P1,P2,P3
1320 FOR Z=1TO500:A$=INKEY$
1330 IF A$=""THENNEXTZ
1340 IF A$="S" THEN GOSUB1370
1345 IF A$="Q" THEN 20
1350 NEXTI
1360 FORZ=1TO2000:NEXT:GOTO50
1370 A$=INKEY$:IFA$=""THEN1370
1380 RETURN
1390 DATA"FOOD","TEXTILES","SIDE ARMS","MEDICINES","CHEMICALS","ORES","URANIUM",
"EXPLOOSIVES","LUXURIES","GEMS","HI-TECHS","NARCOTICS"
1400 DATA"CARGO HOLD",9,"FUELTANK",5,"10 MW LASER",7,"HYPERDRIVE",1,"DRILLING KI
T",1,"AST. DRILL",1,"WELLCAP",1
```

Register and that he can't go any further and that holding down CLEAR and N gives extra lives up to a total of 999. He has asked me to publish his address so that he can enter into communication with other (dare I use the word) hackers but I'm not sure if this is such a good idea as then people might write to him instead of me and I'd be all alone in the world and go slowly mad here in my padded cell. However, I will publish his address, as soon as he sends me his phone number! This is not, you understand for any solicitous reasons, but so that I can clarify some points that have arisen from his letters, although, perhaps we could go out for a quiet drink afterwards, and then back to my penthouse flat for a wild night of loud music, gambling, and, of course, hacking. And we could always have a **aspidistra** too! (One for the editing pen there Helen!). (*This man reads minds — Ed.*)

Sorry, I forgot myself there, some more clues and hints for games, in no particular order, and from a Mr Anart Petel of Coventry.

Crazy Painter: On letting the space invader reach the bottom, when your paint reaches the zero leave it and you'll have an infinite supply. (I can't get this to work, Anart . . . maybe someone else can!)

Hungry Horace: The guard can't get you when you're both in the tunnel so you can pass straight through him.

Tim Love's Cricket: Using a "Quickshot II" joystick, when playing against the computer, position the bowler as far down as you can, then position the bowler's arm in a top-left position and bowl. A large percentage of the time you'll be able to bowl a full toss at the top of the off stump and get it out.

Finally, also from Anart, he says if you load games with the command "AUDIO ON:CLOADM:AUDIO ON" you can insert a music cassette in the player and zap along to, and I quote, "Talking Heads, Simple Minds or Tom-Tom Club". Personally, I find that I play best to Tangerine Dream, Robert Wyatt, Tom Waits (*Swordfish Trombones* and *Rain Dogs* especially) and any early David Bowie. Has anyone found the ultimate record to play along with? If so, you know the address.

Now then, to a topic which has been winding me up for some months now: the heroes in games, and the prejudice involved in picking them. Quite simply, and there is not a hint of a joke here, I mean this: I am pretty fed up with every game's hero being both white and male and getting the girl at the end of the game. I have never managed to impress women by killing off a race of space invaders, I don't think many women are interested in a cosmic space hero as a partner for life! How about a little of sensitivity, how about a black lesbian heroine? How about racism and sexism?

Right, that should start the letters flowing. If you agree or disagree, or you know of games which break the convention, write to me, and together we'll lead the Dragon world to revolution.

Or at least to a little coffee shop I know off Leicester Square. (*Back to square one — Ed.*)

That's about it for this month. Please keep writing, tell me what you'd like to see, stop sending me your high scores, I'm not prepared to massage your egos, and let's try and get the Dragon games crew thinking about what it really wants to play, remember, software writers read this column too!!

Okay, let's do it to them before they do it to us.



I HAVE to admit to being all at sea this month, and the blame for that has to go to Microdeal and their new adventure, *Aquanaut 471*, but more of that later. First to reader James Boyden and a query he's got on an adventure called . . . wait a minute, *Aquanaut 471*? Hang on, James, you might at least wait for me to review them before you start getting stuck in them. James wants to know where the seaweed is. I've half a mind not to tell him. (Those who said I've only half a mind anyway, kindly leave the page.) Have you encountered the Mutant yet, James? If you can get past that beastie then go W-N-E-S TO REACH THE SEAWEED — and then you've got to know how to get it: SRETTUC TNALP HTIW TI TUC.

Syzygy

James also wants to know how to get to the planet on *Syzygy*, the answer to which was given last month, but in case you missed it (how dare you!) you enter the co-ordinates then pull the lever, the co-ordinates for the planet being 0-4-1-5 (and that's the right way round just for a change). Finally on *Vortex Factor*, how do you open the safe? You send an SAE to Tom Wilkinson of 13 Shaftesbury Ave, Hull, Humberside, that's what. No, Tom isn't a safecracker, or at least I don't think he is, but he has just written to me claiming his *Dragon User* merit mark for having solved *Vortex Factor*.

Mr. A. M. Norris of 10 Molescombe, Fairwater, Cwmbran, Gwent NP44 4HE wants to swop or sell *Syzygy*, *The Hulk* and *El Diablero*, if anyone is interested, while Alex Illegible-Signature of Redruth, Cornwall wants to know how to pass the force field on the garden planet. A good job I know he means *Trekboer*, in which protection from the force field is given by TELUMA EHT.

A *Trekboer* tip from P. Sheppard of Cheshunt, for people wondering how to deal with the second spider. The answer: REDIPS DNOCES ON SI EREHT. To make sure of that you must deal with the first spider properly, and take it when dead to MOOR SODNEX-XE EHT, then EDISTUO NOTTUB EHT SSERP. This reader also asks which Dragon adventures will also run on the Tandy computer, and I think it would be a good idea if we could compile a list of those for some future column, so can all Tandy owners out there write in with a list of those Dragon adventures you know for sure will run on the Tandy, for the benefit of everyone.

Yet more help needed on *Trekboer*, this time by Adrian Hall of Huddersfield, and how to stop the Xendos flower from dying is the first question. Don't plant it too early in the game, Adrian, as it's almost the last thing you do. Secondly, how to get the ice to water the flower when necessary: TEKNALB EHT NI TI YRRAC. In *Caverns of Doom*, how to build the raft when you've got the tools, hammer, nails, saw and beams: EPOR EHT DEEN LLITS UOY. And how to explore underwater without running out of air: PU-N-WN-N-N-N-D OG.

If anyone knows how to get over the oil fire on the third level of *The Poseidon Adventure*, can they write and tell Stephen Heraton, 4 Bankcroft, Langton, Preston, Lancs PR4 5AL, and Stephen also wants to know how to obtain food from the dinner droid in *Juxtaposition*, which is straightforward enough: "DOOF EM EVIG" DIORD RENNID OT YAS. You don't even need to say please.

Mr B. W. Le Roux writes to me on his flash Amstrad PCW from April Cottage, Cliff End Lane, Pett Level, Nr Hastings, East Sussex TN 35 4EF, and has come up wtih a potential money-making scheme. He says I should print pages of clues every month but make them coded instead of merely backwards — then I can make a few bob by selling the code-books. Nice thinking, Mr Le Roux, but I wonder if the new editor would let me get away with it? I can't give a clue backwards, forwards, cryptic or in binary for this reader's problem in *Syzygy*, as he's blundered into a big forest where the light is a strange colour and he can't do anything. Can anyone shed any un-strange light on that problem? The same reader is also having light trouble in *Caverns of Doom*, wondering how to mend the broken lamp. He says he's tried using the oil to un-rust it, the rope to improvise a wick, and has even taken the pickles out of the jar to try to put the lamp in there to shield it from draughts, but all to no effect. I'm not surprised, either. Why? EREHWESLE PMAL NEKORBNU NA S'EREHT. Why hasn't this adventurer discovered it yet? It looks from the map as if he hasn't MOOR LLAMS EHT NI LLAW EHT DENIMAXE. Mr Le Roux also offers help on *Pettigrew's Diary*, amongst others, which not many people claim to have finished.

Where is the bomb hidden in *Wings of War*, asks A. Court of Birmingham. RALLEC EHT NI, says I, where you go HTRON TSAE HTUOS HTUOS then RENIATNOC XIF with HCNERW DNA MUINIMULA and finally TSAE OG.

All this backwards writing is taking its toll, so I'll give myself a break by looking at *Aquanaut 471: Under the Doomed Sea*, another of Microdeal's imported American adventures converted for the Dragon, and also available for the Tandy. This takes place in the 21st century, where Jacques Cousteau's vision of underwater cities has become a reality, and you play the part of a high-ranking member of the Oceanic Federation. It seems there's trouble at t'Trident Dome, as that's where you're headed when the adventure begins, though you won't know what the trouble is till you've contacted Huey-14, the Dome's service droid, who sent out the SOS.

You only have a limited number of moves to find the Dome when travelling around the various underwater locations, but luck led me to it first time — bad luck, I think, as this then brings you to the first of what might be called arcade games, if arcades had ever been so primitive. You have to use your joystick to manoeuvre your submarine across the screen through bubbles floating to the surface to reach the Dome's landing spot. It's awkward to do, and rather annoying if, like me, you'd been looking forward to getting stuck into a new adventure. Once you're through, be sure to save your game so you don't have to go through the silly game again.

Underwater city

Here the adventure proper does begin, and you can start to map out the underwater city, wondering what you do with the lead pipe and the memory grid that you soon find. Not that you worry for long as, oh no, it's another 'arcade' game . . . just when it was starting to get interesting, too. This time you move your little self up the screen through a maze past a series of moving robots.

I'm afraid I found these arcade elements tedious and disruptive in what would otherwise be a promising adventure with the high standard of moving graphics we've come to expect from the likes of *Trekboer* and

others. No doubt it will be popular with many of you, but it's not one I intend to load up too often. Let's end the mini-review with a clue, though. What to do with the tool-chest? ECIWT KOOL.

Back to an old favourite, and I don't mean D. Collins of Biggleswade, but the adventure he or she is asking about, *El Diablero*. What do you do with the desert beetles? BMOT EHT NI MEHT PORD. And with the seeds? HTAP NIATNUOM EHT NI MEHT TNALP. And the mat? SDAERHT EHT ENIMAXE. And how to get back from the cave when you've dreamed yourself there? EKAWA. Help is available from this same reader on *Franklin's Tomb, Mansion Adventure, Dragon Mountain, Calixto Island, Don't Panic, Ring of Darkness, Mission 1* and *Mystery of the Java Star*, so SAE if you're interested to D. Collins, 8 High Road, Broom, Biggleswade, Beds SF18 9NJ.

Envelopes

On the subject of stamped addressed envelopes, don't forget to send them if you're writing to anyone asking for help, and that goes if you're writing to me and want a personal reply. No SAE means the query will get dealt with in the next available issue of the magazine, and depending on publication dates that could mean a wait of two or three months for you.

From David Walshaw, a citizen of Wakefield, come several pleas, plus a map and clues on *Dragon Mountain*. To kill a demon in this older adventure: REGGAD RO DROWS CIGAM ESU. And the Elf? NEDLOG GNIHTYNA HTIW DEBIRB EB NAC. Amongst David's problems is the mermaid in *Sea Quest* — well, I don't expect you come across many in Wakefield, do you? So you wouldn't know what she wants: RORRIM EHT.

Anyone got a spare Pi-Man? Michael Higgins of 18 Westland Drive, Glasgow G14 9NT would like to play *Pimania* but can't get hold of a copy, so if you've finished with yours and would like to sell or swap then contact Michael direct. Alistair Grant lives near Droitwich but is also lost in space in *Lost in Space*. He keeps getting caught by the security robot, but even when he manages to make it to the room with the grill he doesn't know what to do when he gets there. Use your head, Alistair, or at least that hole in the front of it that you shovel food down: EVIF RO RUOF EERHT OWT ENO YAS.

A letter has arrived from Michael Edwards, alias Broomsoft, to show that he doesn't just write adventures but solves them as well. He's sent in a complete solution to *Black Sanctum*, so if anyone wants a copy of that just write in to me with SAE. Meanwhile here are a few of the problems dealt with for you:

1) Can't get through the locked door? TFOL EHT OT OG
2) Can't pass caretaker? ENIW EHT PORD
3) Can't find music to play? NIAGA ESACKOOB ENIMAXE

Michael has *Franklin's Tomb, Pirate Adventure* and *Ring of Darkness*, which he'd like to swap for *El Diablero, Shenanigans, Sea Quest, Return of the Ring, Calixto Island* (graphics preferred), *Lost in Space* and *Fishy Business*. Any offers? Write to 30 Broomhills, Welwyn Garden City, Herts AL7 1RF.

Finally a big thanks to Simon Hargrave of Gloucestershire for his solutions to *Juxtapositon* and *Trekboer*. I've already made the last one available to readers, so let's add *Juxtaposition* to the list of freebies, as well as *Black Sanctum*. Don't say we never give you anything! We might even give you another adventure column next month, if you behave yourselves.

Adventure Contact

To help puzzled adventurers further, we are instituting an Adventure Helpline — simply fill in the coupon below, stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpline, 12/13 Little Newport Street, London WC2H 7PP. As soon as enough entries have arrived, we will start printing them in the magazine.

Don't worry — you'll still have Adventure Trail to write to as well!

Adventure ....
Problem..
....
Name ....
Address..
....

Adventure Contact

Adventure: Vortex Factor. **Problem:** How do I open the safe? How do I unlock the door at the North? How do I get the Time Machine to work? **Name:** John Foster. **Address:** 94 The Oval, Firth Park, Sheffield S5 6SP.

Adventure: Juxtaposition. Problem: Where is the Altarian Orchid and what is the use of the seed? **Name:** Paul Davidson. **Address:** 211 Dunminning Road, Glarryford, Ballymena, Co. Antrim, N.I. BT44 9PP.

Adventure: Juxtaposition. **Problem:** How the *** do you get past the Nighteye Droid? **Name:** Bob Riding. **Address:** 24 Wittmills Oak, Buckingham, Bucks MK18 7BH.

Adventure: 1) Strange Oddysey. 2) Escape from Pulsar 7. 3) El Diablero. **Problem:** 1) What use is the Rigilian Dia-Ice Hound? Does the pick-axe help? 2) How do I mend the lathe? How do I open the locker in the maze? How do I make a cake? 3) Where do I find the twig? I have already spoken to the lizard. **Name:** Robin Hemmings. **Address:** 8 Ingleby Road, Wigston, Leicester LE8 1DQ.

Adventure: Return of the Ring. Problem: Can't get books of skulls from Mutant in forest moon. **Name:** Mark Ryan. **Address:** 5 Sully Terrace, Penarth CF6 2DS.

Adventure: Syzygy. **Problem:** I can't find the teletransporter. I can't kill Vader. **Name:** Cristina Garcia-Verdugo. **Address:** c/Illescas No. 145, 8°B 28024 Madrid, Espana.

Adventure: Trekboer. **Problem:** How do you get to the planet's surface? How do you see the dark room and how do you open the access panel? **Name:** Paul Everitt. **Address:** 1 Windmill Road, North Anston, Nr. Sheffield S31 7EH.

Adventure: El Diablero. **Problem:** I have been everywhere I can, but I can't do anything. Help! **Name:** Jason Coomes. **Address:** 52 Springfield Avenue, Holbury, Southampton SO4 1LP.

Adventure: Return of the Ring. **Problem:** How do I kill the Genie and the Trog? **Name:** Roger Pigott. **Address:** 27 Welbeck Road, Walkley, Sheffield S6 5AY.

Adventure: Juxtaposition. **Problem:** Where do I find the red ore? How do you get to move N,S in Power Pyramid? Where do I find the space? How do I cross the river into Baron Blue's land? I need a hint sheet and a map. **Name:** Michael Pointing. **Address:** 82 Raymond Road, Bedminster, Bristol BS3 4QW.

Adventure: Juxtaposition. **Problem:** Cannot find the mask. **Name:** James Woollard. **Address:** 6 Boxford Court, Haverhill, Suffolk.

Adventure: Return of the Ring. **Problem:** How do you breathe on moon forest? How do you get hit points and more passes? **Name:** Karl Lawson. **Address:** 8 Walworth Grove, Primrose, Jarrow, Tyne & Wear NE32 5YP.

Adventure: Jerusalem Adventure 2. **Problem:** How do I get through the golden gate? What are the magic words? **Name:** Julian Griffin. **Address:** 1 Higgs Close, Rowlatts Hill, Leicester LE5 4LY.

Adventure: Shenanigans. **Problem:** I can't get the pole into the cave. **Name:** Mark MacMillan. **Address:** 22 Trinity Place, Deal, Kent CT4 9HH.

Adventure: Temple of Vran/The Final Mission. **Problem:** Can't get past the security guard, or through the double doors. **Name:** Ian Howie. **Address:** 20 Hallfield Road, Aberdeen AB2 6RQ.

Mini Maths

***Gordon Lee** sets the puzzles — but who will find the totals?*

FROM TIME to time on this page we present a series of unrelated puzzles that the reader might be interested in tackling, purely for his (or her) own enjoyment. Here are five such puzzles, but the solver should proceed with caution as not all of them are necessarily soluble by the use of the computer.

1. Using each of the nine digits 1 to 9, once and once only, arrange them to form a perfect square. For example, one such arrangement might be 382945761, the square of 19569. There are many other arrangements of the nine digits possible, but can you find a) the lowest possible number, and b) the highest?

2. As in question 1, use the nine digits 1 to 9, but this time arrange them to form two nine-digit *prime* numbers: a) the lowest possible, and b) the highest.

3. A farmer has a circular field with a diameter of 200 feet. The field is enclosed by a fence. Inside this field tied to the perimeter fence is a goat on a length of rope. If the rope is long enough to allow the goat to graze exactly half the area of the circular field, how long in the piece of rope? (You should assume that the goat can reach exactly to the end of its tether).

4. The number 4 is very interesting. It is a perfect square, and the two integars each side of it, 3 and 5, are *both* prime. Can you find the next highest square number that has this property?

5. Consider the following alphamatic:

$$\frac{\text{DRAGON}}{\text{USER}} = \star\,\star$$

In alphamatics, each letter represents a certain digit, wherever it occurs. Similarly, unlike letters represent unlike digits. So in the puzzle given we have a six-digit number with all digits different (as represented by the word 'DRAGON'), divided by a four-digit number, also with different digits ('USER'). Note however that the second digit of the numerator is the same as the final digit of the denominator, as represented by the letter 'R'. The result of this division is represented by the two asterisks. Of this number nothing is known except that it consists of two digits, which may, or may not, be alike. However, if this value is cubed, and the digits of this cube exchanged for letters using the same substitution as in 'Dragon User', the result will be a familiar English word. Can you find the correct values of the letters and the word so produced?

For the competition this month, we require *just* the solution to this final question (No. 5).

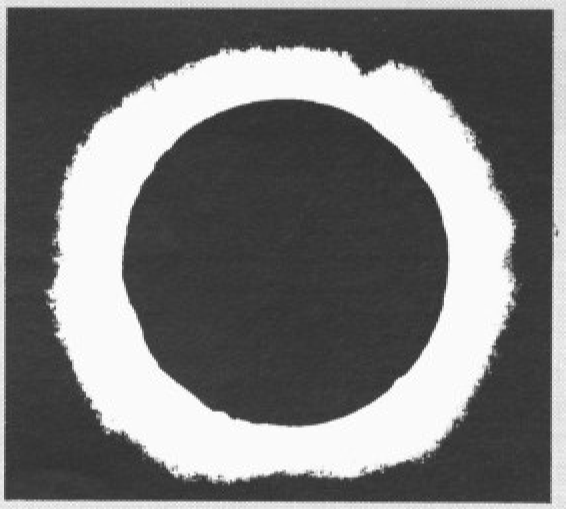
The solution to question 1 follows, so if you wish to tackle it yourself, read no further. The solutions to questions 2 to 4 will be published next month.

1. The lowest and highest numbers are 139854276 and 923187456, the squares of 11826 and 30384 respectively. One method of solution is by running the program listed below. Here, the values in the range 11112 to 31426 are squared and the resulting figure is tested to determine if it is comprised of the nine digits, 1 to 9. In order to do this, the numerical variable needs to be converted into a string variable so that individual digits can be selected for examination. Note the second command in line 30: S$=MID$(S$,2) which removes the first character of the string. On the 'Dragon', as on some other micros, when a numeric variable is converted to its string equivalent, an extra 'invisible' character appears in the first position in the string. This character holds the imaginary plus sign in front of all positive numbers. (If the number were negative, the minus sign would appear here, as we would expect). Consequently, we need to remove this extra character from the string before examining its contents.

Having done this, the remaining nine digits need to be tested to determine that there is no zero present, and that no digit is duplicated. The test for duplication is not difficult as every character in the string can be compared with every other, but it is time-consuming to test every one of the possible squares in this way. Fortunately there is a short cut available. In any arrangement of the digits 1 to 9, the sum of the digits must equal 45, and the product must equal 362880. Consequently, it is simply necessary to move along the string taking each digit in turn and finding the respective sum and product. Any value not in agreement with the expected values can be rejected. Note that this procedure will not guarantee that every number which passes the test does contain the nine digits, but it will reject all those which do not have sums of 45 and products of 362880. If the listed program is run it will print out all 30 squares which are made up of the digits 1 to 9, but it also includes a further two containing duplicated digits. However, these are easily spotted and can be eliminated. Note that the question, as stated, only asked for the lowest and highest values, so if it is run until the first answer is printed (the lowest value), and then line 10 is amended to: 10 FOR N=31426 TO 11112 STEP −1, the program can be re-run to compute the highest value.

Prize

Guess what we've got this month? A real turn-up for the bookmakers. For those of you who reckoned your chance of getting a bug-free copy of *Total Eclipse* were (speed of light) to one against, Eclipse-Fenmar are backing their complete confidence in the



1.3 version by sending us here 20 copies for our August prizewinners. Our man on the spot raved about *Total Eclipse* . . . fleets of Traders raved in frustration when the Universe locked up a thousand light years from home. Now it seems that the Universe is rolling again.

Rules

To solve this puzzle and win a prize, you must give the answer and show how you arrive at it, using a Basic program. Don't send cassettes, just a printout, and any explanation you want to offer (cat and four kids to support won't be accepted). Please mark your envelope AUGUST COMPETITION, and don't forget to put your name and address on your entry and send it to reach us by mid-September at the latest.

This month's tiebreaker, complete the following phrase: "You're never alone in the Universe . . ." It can be a limerick if you insist — just so long as it begins "You're never alone in the Universe . . ."

May Winners

Our trusty trophy drawer this month contains 20 copies of Blaby's hit *Kung-Fu — The Master,* and these are on their way to J. B. Slinger of High Wycombe, Andy Beale of Wallington, Phil Sapiro of Liverpool, E. C. Hastead of Erith, D. C. Lee (not *the* Dee C. Lee) of Barnsley, J. L. Clark of Portsmouth, Ralph E. Newman of Pentridge, S. P. Holt of Rosyth, P. D. Maddocks of Taplow, Simon Aubrey of Swindon, G. A. Hunt of Carnforth, G. R. Barber of Sutton Coldfield, E. A. Newman of Addlestone, M. W. Stonton of Towcester, Mark Heaps of Washington, P. A. Bennington of Strood, P. Fairbairn of Kilmaurs, M. C. Regnier of Fetcham, E. V. Jones of Cardiff and D. C. Faulkner of Pontypridd.

Solution

Many calculated correctly that 6 to the power of 36 is 103144247984905355 46171949056 . . . but not everyone remembered to say that the odds involved are *either* 1 in . . . 056 *or* . . . 055 to 1. Tiebreakers 'drew' heavily on unearthed Dragons, but the Editor's favourite is 'I get a kick out of my Dragon, because it's perfect for whatever ju do'. Ouch!

COMPUSENSE SUMMER SALE

DASM/DEMON Cartridge **£14.95**
EDIT + Cartridge **£14.95**

BARGAIN PACK for DRAGON DOS/CUMANA DOS

DASM/DEMON DYNAFAST/DYNAMISER/
DYNAXREF HIRES/EDIT+ all for **£24.95**

SPECIAL BARGAIN PACKS for DRAGON FLEX

FLEX/EDITOR/ASSEMBLER/DBASIC/
DYNACALC/RMS/SP-EDIT **£129.00**
DYNACALC/RMS/S-EDIT **£75.00**
FLEX UTILITIES/DEBUG/
DIAGNOSTICS **£75.00**

PROFESSIONAL BUSINESS SOFTWARE for OS-9
THE MOST COMPREHENSIVE PACKAGE AVAILABLE FOR THE DRAGON

As originally commissioned by DRAGON DATA
Volume 1: Invoicing/Stock Control
Volume 2: Sales Ledger/Purchase Ledger/Labelling
now dramatically reduced to only **£29.95** each
or **£54.95** for both

DRAGON HARDWARE

4-way expansion unit **£125.00**
EPROM programmer **£75.00**
RS232 — I/O module **£49.50**
Sideways ROM **£39.00**
Disk Drives for your DRAGON — check with us for the best prices — free diskettes with every disk drive

5¼in diskettes (bulk pack) 10 for **£10.00**
50 for **£45.00** 100 for **£85.00**

CHAT COMBO — the complete modem package for DRAGON FLEX/DRAGON 64, CHATPLUS2 + CHAT/Viewdata + XMODEM. **£45.00**

Modems: DEMON (not BT approved) **£69.95**
WS2000 (BT approved) **£125.00**

DEMON Modem with cable and CHAT COMBO **£99.95**
WS2000 modem with cable and CHAT COMBO **£159.95**

DRAGONPLUS — the Ultimate Expansion for your DRAGON — 80 by 24 video — extra 64k memory **£103.50**
EDIT+ for DRAGON DOS now includes software for DRAGONPLUS
FLEX update disk **£5.57** OS-9 Drivers **£5.75**

Prices include VAT
PLEASE ADD £2.00 to all orders for Postage and Packing
VISA/ACCESS/MASTERCHARGE credit cards accepted

Stocks of some items are limited — order now to avoid disappointment

COMPUSENSE Ltd

PO Box 169, Palmers Green
London N13 5XA

Fast mail order service
Telephone orders on 01-882 0681/6936

PEAKSOFT DRAGON SUPPLY CENTRE

CHEETAH 125 AUTOFIRE JOYSTICK with DRAGONISER £14.95. FOUR (!!) fire buttons plus autofire! We've chosen the Cheetah as one of the best value-for-money arcade joysticks available. DRAGONISER only (supply your own joystick) £11.95. Unlike many conversions, our unique Peaksoft Dragoniser does NOT invalidate the joystick manufacturer's guarantee.

PRO-STICKS £9.95 pair, £5.50 each. Genuine twin potentiometer floating joysticks, suitable for all purposes. We send these all over the world!

★★★NEW★★★ Single-sided double density DISK DRIVE UNITS as supplied to Dragon Data, without cases or power supplies. Just "discovered" in store by us! Supplied untested at £39.95!!!!!!! Special order form needed — please send self-addressed envelope.

PRISM MODEM

Enables you to join the fun on Radio Dragon (now on the air 30 hours a week!) Micronet, Prestel, and 1200/75 bulletin boards!

There are now FOUR HUNDRED THOUSAND pages to explore on Micronet and Prestel alone, offering inter-active games, nationwide "chat" lines, advice on computer problems, tele-shopping, a micro market, electronic bank accounts, your own free mailbox . . . the list just goes on and on.

The Prism Modem uses your phone line to link your Dragon 32 or 64 to mainframe computers throughout the country — the special cheap call rate costs just 40p an hour.

OUR BREAKTHROUGH PRICE IS JUST £64.95 complete with all leads and "plug-in-and-go" cartridge software, and including a FREE 3-month subscription to Micronet and a FREE copy of the Dragon User's Communications Directory.

ONE DAY EVERY HOME WILL HAVE ONE — WE'RE OFFERING YOU THE FUTURE!

xxxxxx

PEAKSOFT BRAND NEW HEAVY DUTY POWER SUPPLIES £14.95 complete. Compare this with the price of "reconditioned" units!

DRAGON DATA DISK DRIVE COPPER-BOUND TRANSFORMERS £14.95. We tracked down the last in the country and bought the lot! Safeguard the future use of your disk drive while stocks last!

★★★NEW★★★ DRAGON SWEAT SHIRTS (RRP £7.50) Our price £4.99!!! T-SHIRTS (RRP £3.50) Our price £2.49!!! Please state S, M, L or XL size. HALF-PRICE SPECIAL OFFER — For EACH £10 spent on other goods, order ONE sweat shirt for £2.49 or ONE T-shirt for £1.24!!! HOW DO WE DO IT?

★★★NEW★★★ Briefcase-style padded case for Dragon and power supply £9.95!

★★★NEW★★★ Dragon keyboards £14.95 **★★★NEW★★★**
★★★NEW★★★ I LOVE MY DRAGON car sticker 99p **★★★NEW★★★**

LEADS: cassette £2.50, aerial £2.50, printer £11.95. Heavy duty DUST COVERS with Dragon logo £3.95.

Dragon-compatible Omega DATA RECORDER £24.95.

PRINTERS from £194.95 (Tatung TP100) — or our new SUPER DEAL saves £50 on Panasonic's terrific Near Letter Quality P1080, complete with a 1K buffer — now £219. Lead £5 when purchased with printer.

Fidelity colour MONITOR (it's RGB AND Composite, so you won't need to buy a new one if you swap computers!) £199.95 — optional 4-year tube guarantee.

Ask for our complete DISC DRIVE prices!

Plus . . . Dragon User back numbers at £1 each and thousands of the best Dragon books at crazy prices! (Please phone or send sae for details.) For fact sheet on Prism Modem, printers, monitor, or our information sheet "Choosing a Joystick", please send sae.

To order: Just note your name, address, computer and goods required on the back of your cheque or postal order. Access/Visa card-holders may phone their order to 0636 705230 at any time, day or night. All goods are POST FREE.

We supply by return of post, subject to availability. Personal callers welcome — please phone first so that we can have your order ready. CASH ON DELIVERY on all orders over £40 — please send at least £5 deposit and pay the balance to the postman (sums over £50 must be paid at your local Post Office.) Free credit to personal callers by post-dated cheques.

PEAKSOFT

48 QUEEN STREET, BALDERTON
NEWARK, NOTTS NG24 3NS
Tel: 0636 705230

MAKING YOU GLAD YOU BOUGHT A DRAGON!